THE USENIX ASSOCIATION NEWSLETTER

;login:

JUNE 1996 • VOLUME 21 • NUMBER 3

CONTENTS

The UNIX and Advanced Computing Systems Professional and Technical Association

;login: is the official newsletter of the USENIX Association.

- *;login:* (ISSN 1044-6397) Volume 21, Number 3 (June 1996) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.
- $25 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year.
- Second-class postage paid at Berkeley, CA and additional offices.
- POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

EDITORIAL STAFF

- Rob Kolstad, Editor <*kolstad@usenix.org*>
- Tina Darmohray, SAGE News Editor <*tmd@usenix.org*>
- Nick Stoughton, Standards Report Editor <*nick@usenix.org*>
- Carolyn S. Carr, Managing Editor <*carolyn@usenix.org*>
- Sylvia Stein Wright, Copy Editor
- Rhea Gossett, Proofreader

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
Fax: 510 548 5738
Email: <*office@usenix.org*>
WWW URL: <*http://www.usenix.org*>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, and announcements to *;login:*. Send them via email to <*login@usenix.org*> or through the postal system to the Association office. Send SAGE material to <*tmd@usenix.org*>. The Association reserves the right to edit submitted material. Any reproduction of this newsletter in its entirety or in part requires the permission of the Association and the author(s).

;login: is produced with Framemaker 5.0 software provided by Frame Technology and displayed on an NCD X Terminal, donated by Network Computing Devices. The system is served by a Sun SPARCsystem 10 server. Final output is created by a 600 dpi QMS 860 laser printer, donated by Quality Micro Systems. The newsletter is printed on recycled paper.

The closing dates for submissions to the next two issues of *;login:* are June 12 and August 14, 1996.

S A V E T H E S E D A T E S !

Upcoming USENIX Events

2nd Conference on Object-Oriented Technologies and Systems (COOTS)
June 17–21, 1996, Toronto, Canada
Program Chair: Douglas C. Schmidt, *Washington University*
Tutorial Program Chair: Doug Lea, *SUNY Oswego*

4th Annual Tcl/Tk Workshop '96
July 10–13, 1996, Doubletree Hotel, Monterey, California
Program Chairs: Mark Diekhans, *The Santa Cruz Operation, Inc.;* Mark Roseman, *University of Calgary*

6th UNIX Security Symposium—Focusing on Applications of Cryptography
July 22–25, 1996, Fairmont Hotel, San Jose, California
Sponsored by the USENIX Association. Co-sponsored by UniForum in cooperation with The Computer Emergency Response Team (CERT)
Program Chair: Greg Rose, *RoSecure*
Camera-ready papers due: June 10, 1996

10th Systems Administration Conference (LISA '96)
September 29–October 4, 1996, Chicago Marriott, Chicago, Illinois
Co-sponsored by USENIX and SAGE, the System Administrators Guild
Program Chairs: Helen Harrison, *SAS Institute;* Amy Kreiling, *University of North Carolina.*
Invited Talks Co-ordinators: Rik Farrow, *Internet Security Consulting;* Kimberly Trudel, *Massachusetts Institute of Technology*
Notification to Authors: June 11, 1996; Final Papers Due: August 15, 1996

2nd Symposium on Operating Systems Design and Implementation (OSDI '96)
October 28–31, 1996, Seattle, Washington
Co-sponsored by ACM SIGOPS and IEEE TCOS
Program Chairs: Karin Petersen, *Xerox PARC;* Willy Zwaenepoel, *Rice University*
Notification to Authors: July 30, 1996; Revised Papers Due for Shepherding: August 19, 1996; Camera-ready Full Papers Due: September 16, 1996

2nd USENIX Workshop on Electronic Commerce
November 18-20 1996, Claremont Hotel, Oakland, California
Program Chair: Doug Tygar, *Carnegie Mellon University*
Extended Abstracts Due: July 16, 1996; Notification to Authors: August 5, 1996
Camera-ready Final Papers Due: October 7, 1996

USENIX 1997 Annual Technical Conference
January 6-10, 1997, Anaheim Marriott, Anaheim, California
Program Chair: John Kohl, *Atria Software*
Invited Talks Coordinators: Mary Baker, *Stanford;* Berry Kercheval, *Xerox PARC*
Manuscripts Due: June 18, 1996; Notification to Authors: August 7, 1996
Camera-ready Papers Due: November 13, 1996

USELINUX: Linux Applications Development and Deployment Conference
January 6-10, 1997, Anaheim Marriott, Anaheim, California
Co-located with the USENIX Annual Technical Conference
Conference Chair: Michael K. Johnson, *Linux Journal*
Proposals due: July 1, 1996

For more information about USENIX and its events, access the USENIX Resource Center on the World Wide Web. The URL is http://www.usenix.org.

FROM THE EDITOR

Another Revelation

I keep thinking that sooner or later I will achieve that state of being that I was promised at high school graduation: "You are now ready to go out into the world." I'm hoping it will be sooner than later.

This month's revelation? It's all about marketing.

For the last dozen or so years, I have worked with the Jet Propulsion Laboratory Explorer Post (Explorers are like bigger boy scouts, except they're really not that much bigger, half are women, and there isn't much scouting involved) and two wonderful individuals from Rockwell International to hold an annual "Space Settlement Design Competition" for pre-college students. This year, the event attracted over 140 youth who represented a diverse cross-section of the students from California. A dozen members of the Post along with another fifteen members of the southern California aerospace technical community run operations and act as both technical and managerial "advisors."

The competitors think that the weekend is all about designing space settlements. This year's competition (set in the year 2036) required the design of a space station (and infrastructure) to house 12,000 people in the orbit of Mars. The competitors produced all sorts of fanciful designs for robots, space stations, and budgets.

In fact, the weekend offers most competitors their first shot at trying to solve a problem that is far bigger than either they (or their instructors) can solve individually before the deadline. The other two founders (Anita Gale and Dick Edwards) and I do our best to set up the weekend so that the competitors are exposed to the fundamentals of corporate teamwork and management. We believe it is one of the only practical examples most of the competitors will see before they actually join the work force (which would be sometime during the years 2000 to 2004 or so, for this year's crop of competitors).

Imagine my surprise as I was listening to the design presentations and realized that these teams really need not just good engineering management but also good marketing.

For years I've wondered about marketing. It consumes so much money and yet the results are so hard to measure. I think I finally have a handle on it: marketing has everything to do with creating a desire in people to buy (own) your product. How hard could it have been over all these years to figure that out? Would you rather purchase "processed cow fluid" or "Ben & Jerry's Gourmet Ice Cream"?

So next year, we'll try to add a marketing division to the simulated companies. The marketing division will try to find simple, appealing messages to send to potential consumers (the judges). These messages will present all the good ideas in the technical designs without too many of the details that competitors generally have difficulty in describing.

I'll be looking at my company's marketing in a more enlightened manner as well.

RK

USENIX Members Benefits

As a member of the USENIX Association, you receive the following benefits:

- Free subscription to *;login:*, technical features, system administration tips and techniques, international calendar of events, SAGE News, book and software reviews, summaries of sessions at USENIX conferences, Snitch Reports from the USENIX representative and others on various ANSI, IEEE, and ISO standards efforts, and much more.
- Free subscription to *Computing Systems*, the refereed technical quarterly published with The MIT Press.
- Access to papers from the USENIX Conference and Symposia, starting with 1993, via the USENIX Online Library on the World Wide Web *<http://www.usenix.org>*.
- Discounts on registration fees for the annual, multi-topic technical conference, the System Administration conference (LISA), and the various single-topic symposia addressing topics such as object-oriented technologies, security, operating systems, electronic commerce, and mobile computing – as many as seven technical meetings every year.
- Discounts on the purchase of proceedings from USENIX conferences and symposia and other technical publications.
- Discount on BSDI, Inc. products. BSDI information: 800 800 4BSD.
- Discount on the five volume set of 4.4BSD manuals plus CD-ROM published by O'Reilly & Associates, Inc. (800 998 9938) and USENIX.
- Discount on all publications and software from Prime Time Freeware, including Prime Time Freeware for Unix, Prime Time Freeware for AI, Prime Time TeXcetera and Tools & Toys for UnixWare. Contact *<ptf@ptf.com>*
- Savings (10-20%) on selected titles from McGraw-Hill (212 512 2000), The MIT Press (800 356 0343), Prentice Hall (201 592 2657), John Wiley & Sons (212 850 6789), and O'Reilly & Associates (800 998 9938).
- Special subscription rates to the periodicals *The Linux Journal* (206 527 3385), *UniForum Monthly*, *UniNews*, and the annual *UniForum Open Systems Products Directory (800 255 5620).*
- The right to vote on matters affecting the Association, its bylaws, election of its directors and officers.
- The right to join SAGE, the System Administrators Guild.

To become a member or receive information regarding your membership status or benefits, please contact *<office@usenix.org>*.
Phone: 510 528 8649.

USENIX NEWS

Election Results

by Ellie Young, USENIX Executive Director
<ellie@usenix.org>

The results of the elections for Board of Directors of the USENIX Association for the 1996-1998 term are as follows (* = elected to the Board).

President:

Andrew Hume*	992
abstain	70

Vice President:

Dan Geer*	973
abstain	89

Treasurer:

Eric Allman*	1019
abstain	47

Secretary:

Lori Grob*	661
Peter Collinson	364

Directors:

Elizabeth Zwicky*	738
Margo Seltzer*	665
Peter Honeyman*	649
Greg Rose*	612
Pat Parseghian	539
Ed DeHart	391
Doug Kingston	342

Total number of ballots cast: 1082
Total number of invalid ballots: 1

The newly elected Board's term will begin following the next regularly scheduled USENIX Board meeting, which will be held May 17-18, 1996, in Fairfax, VA.

Farewell

by Steve Johnson
<scj@mathworks.com>

As I leave the USENIX board, tradition suggests that I should write a mushy piece looking backwards on my tenure on the board (since 1984, with a two-year hiatus). Although the changes have been dramatic, and in some ways interesting, I find the traditional piece very hard to write. I find myself much more struck by how much things are the same, rather than how much they have changed.

One thing that is the same is that, almost continually since 1984, people have predicted the financial collapse of the "pure" UNIX vendors. The competitors have varied – from AIX to Apple to Xenix to NT – but the story has remained very much the same. UNIX vendors are arrogant techies who don't understand marketing and who can't get their act together. The UNIX market is fragmented and too small to be interesting. "There's no software," etc., etc.

Now some of the "pure" UNIX vendors have gotten out of the business (most notably, AT&T!). And, indeed, many are arrogant and don't have their act together. The UNIX market is small. But there are fewer people going after it, and it seems in many ways less competitive than the PC market. But in the last analysis, none of this matters.

For what it is worth, UNIX is the place where a large number of the technical innovations of the 1980s and 1990s have happened first. From NFS to email to Netnews to Xwindows to Tcl/Tk and Perl to the World Wide Web to Java, we are where it's at and have been for over two decades. And USENIX is where a lot of the technical refining of these ideas has taken place.

Another thing that has stayed amazingly constant is the people. From time to time, we have been worried that we would be overrun by "the suits." But we have successfully repelled all invaders (although I was astounded at a recent trade show to see IBM and AT&T booth representatives in T-shirts, while the UUNet folks were in suits). USENIX meetings are still attended by the breaking edge people in software and systems, but are informal enough that the novices can meet and talk with the more experienced. The meetings have increased in size and have become more diverse (the most visible change is more women), but are still fun, thought provoking, and above all practical.

Another thing that has not changed is our "niche." We remain a bridge between research and the practical, ruled by neither, and visibly noncommercial. Compared to the other groups offering conferences and seminars, we have managed to steer the thin line between academic self-congratulation and crass commercialism. If occasionally we indulge in crass self-congratulation and academic commercialism, at least we get it right most of the time.

SAGE is now roughly 50% of the membership of USENIX, although there are many previous members of USENIX who are now also SAGE members. Although a traditional group of academic USENIX members (the "let's write papers and publish them so that we won't be denied tenure because we are software people" crowd) is underrepresented in SAGE, other traditional groups of USENIX members (particularly the more pragmatic and "systems in the large" types) are very comfortable at system administation meetings. I am delighted that we were able to fill this need – SAGE has invigorated USENIX and will provide a new generation of leaders. And USENIX has strengthened SAGE, providing it with a broader perspective and encouraging people to generalize and share their experiences. I see SAGE more as an evolution of USENIX than a real change.

The final constant is the health and efficiency of the organization. We have continually run a surplus through most of the years I was on the board. We have been able to use this surplus to expand our support for students and other good works. Our staff has been a delight – competent, enthusiastic, and above all small, allowing us to ride through the "bad" years and exploit the good ones. I am continually amazed by the productivity of our staff compared to other similar organizations.

Several years ago, in one of these letters, I urged us not to give up hope (I think the "monster beneath the bed" was NT in this case) because history has shown that technical strength and innovation will continue, will provide enormous business opportunities, and, like moss cracking rocks, will manage to infiltrate the IBMs and Microsofts of the world. Since I wrote that piece, we have seen the Internet, World Wide Web, and Java sweep the world, and many of us are exploiting the enormous business opportunities therein. I told you so . . . .

But these aren't the last technical opportunities – Java isn't the last computer language. I would like to think that the next revolutions will, to a considerable degree, be nurtured in the technical sessions, tutorials, BOFs, and hallways of future USENIX events. Thank you all for electing me to a front row seat at the greatest technical show on Earth, and then running into the ring and putting on such a great show.

1995 Financial Statements

STATEMENT OF REVENUE AND EXPENSES AND CHANGES IN FUND BALANCE
For the Years Ending November 30, 1995 & 1994

REVENUE	1995	1994
Membership Dues	$ 448,969	$ 381,058
Product Sales	99,625	160,844
Conferences	2,536,649	2,451,258
SAGE	124,522	80,846
Interest, dividends & capital gains	161,485	60,892
Other	9,344	28,426
Total Revenue	**$ 3,380,594**	**$ 3,163,324**

EXPENSES	1995	1994
Membership Services/General Admin.	$ 725,263	$ 673,469
Conferences	1,595,055	1,896,964
SAGE Expenses	37,923	32,187
Newsletter & Journal	202,403	185,409
Products	36,675	57,272
Projects	87,695	85,997
Depreciation	17,982	18,606
Total Expenses	**$ 2,702,996**	**$ 2,949,904**
Excess of Revenue Over Expenses	**677,598**	**213,420**
Restricted Funds Received		**121**
Fund Balance Beginning of Year	**2,237,756**	**2,024,215**
Fund Balance End of Year	**2,915,354**	**2,237,756**
Unrecognized Gain on Securities	**329,958**	**(20,138)**
Total	**3,245,312**	**2,217,618**

BALANCE SHEET
As of November 30, 1995 & 1994

	1995	1994
ASSETS		
Current Assets		
Cash	$ 902,473	875,588
Receivables	30,208	72,935
Prepaid Expenses	148,608	112,310
Inventory	36,973	30,724
Total Current Assets	$ 1,118,262	$ 1,091,557
Fixed Assets		
Investment in Securities	$ 2,270,463	$ 1,223,734
Property & Equipment, Net of accumulated depreciation	77,249	56,104
Total Fixed Assets	$ 2,347,712	$ 1,279,838
Total Assets	**$ 3,465,974**	**$ 2,371,395**
LIABILITIES & FUND BALANCE		
Current Liabilities		
Accrued Expenses	$ 84,447	$ 93,498
Deferred Revenue	136,215	60,279
Total Liabilities	**$ 220,662**	**$ 153,777**
FUND BALANCE	**3,245,312**	**2,217,618**
TOTAL LIABILITIES & FUND BALANCE	**$ 3,465,974**	**$ 2,371,395**

STATEMENT OF CASH FLOWS
For the Years Ending November 30, 1995 & 1994

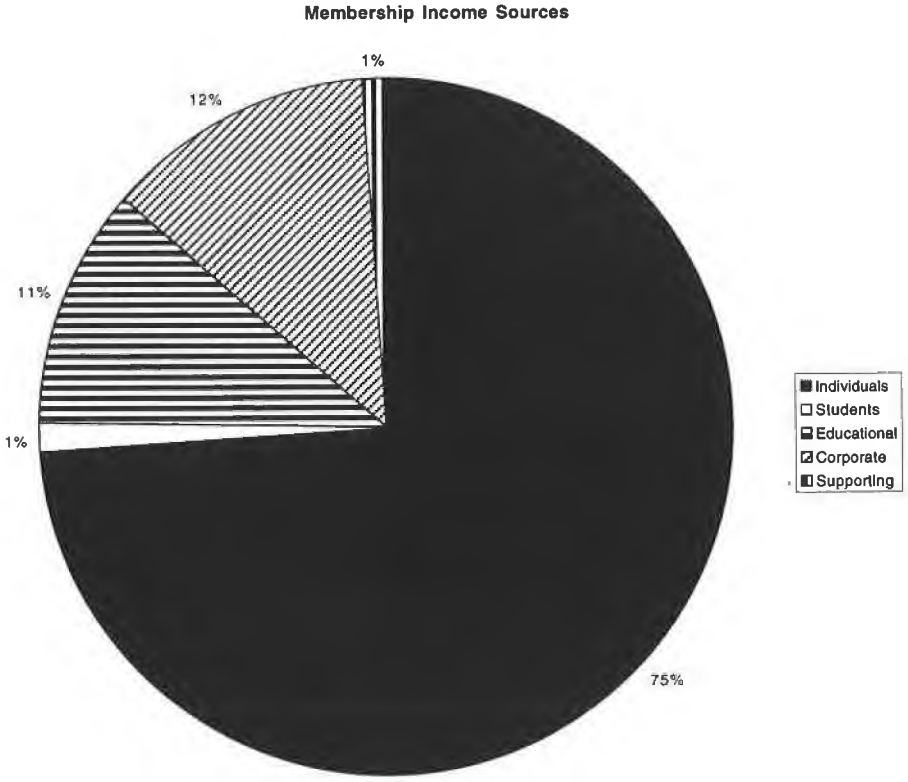
	1995	1994
Excess of Revenue Over Expense	**$ 677,598**	**$ 213,420**
Depreciation	17,982	18,606
Decrease/Increase in Receivables	42,727	(60,703)
Increase/Decrease in Inventory	(6,249)	13,955
Increase in Prepaid Expenses	(36,298)	(23,812)
Decrease/Increase in Accrued Expenses	(9,051)	45,261
Increase in Deferred Revenue	75,936	40,054
Total	**$ 85,047**	**$ 33,361**
Total Net Operating Cash	**$ 762,645**	**$ 246,781**
Receipt of Restricted Funds	$ 0	$ 121
Increase in Long Term Securities	$ (696,633)	$ (39,854)
Increase of Property & Equipment	(39,127)	(31,996)
Total	**$ (735,760)**	**$ (71,729)**
Net Change in Cash	**$ 26,885**	**$ 175,052**
Cash & Equivalents, Beginning of Year	**875,588**	**700536**
Cash & Equivalents, End of Year	**902,473**	**875,588**

Member Dues

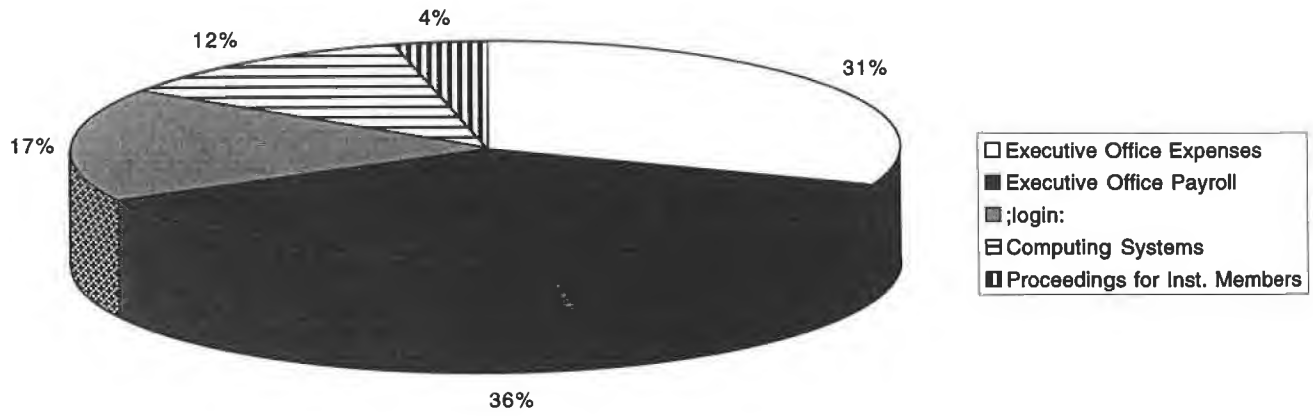
by Ellie Young, Executive Director
<ellie@usenix.org>

Are you ever curious to know how your USENIX and SAGE membership dues are spent? If so, here are a few charts that might help. The first shows sources of USENIX membership dues income, which totaled $449,000 in 1995. The second chart shows how that money was disbursed. Note that the Conference Office does not receive any monies from membership; it is totally funded by income generated from the conferences. You should also know that just one half of the Executive office payroll, general expenses, newsletter and journal expenses are covered by membership dues. The balance is funded by income from conferences and from publications sales. The third chart shows how the Executive Office spends its money. The "Other" category includes items such as taxes and licenses, bank service charges, and miscellaneous expenses for office equipment and maintenance.

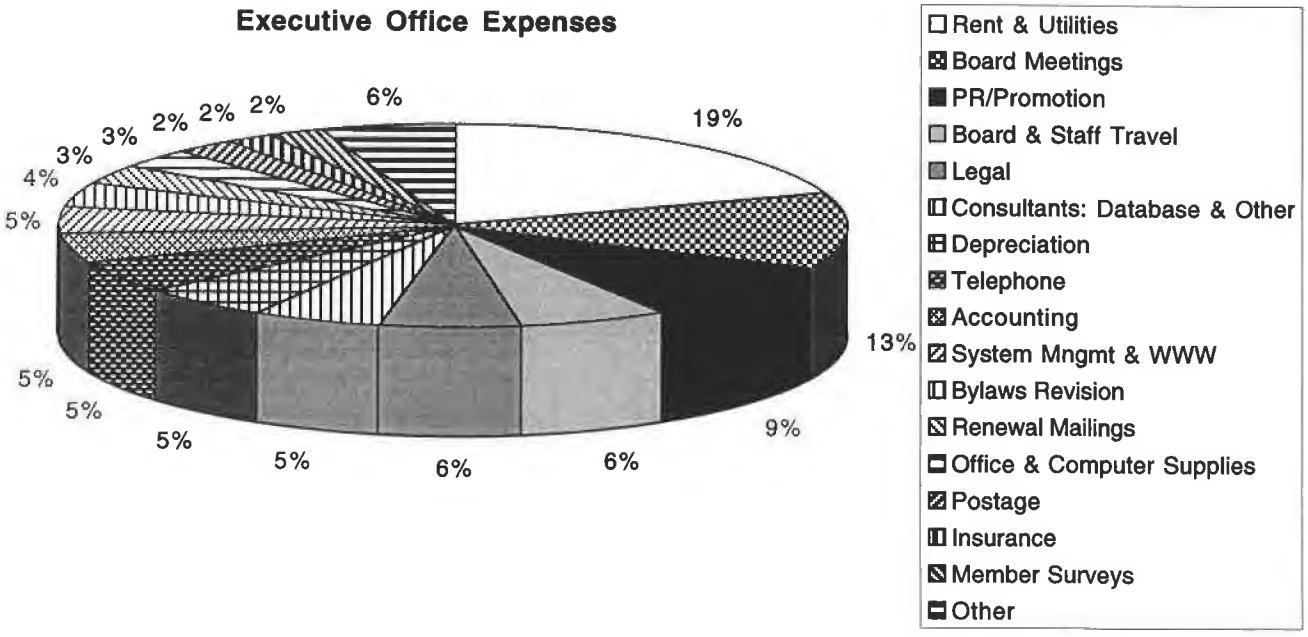
The first SAGE chart shows the sources of all income, which totaled $125,000 in 1995. One-third of the income came from arrangements in which 1) SAGE is a co-sponsor of the SANS conference, and 2) SAGE shares with USENIX and UniForum the net proceeds from sponsoring a set of seminars organized for the UniForum Conference. The second chart shows how that money was parceled out. In 1995, SAGE income covered all of its total expenses (both direct and allocated expenses for administration/payroll). Allocated expenses (staff and overhead) are not reflected in the charts and are currently 50% of the entire SAGE budget.



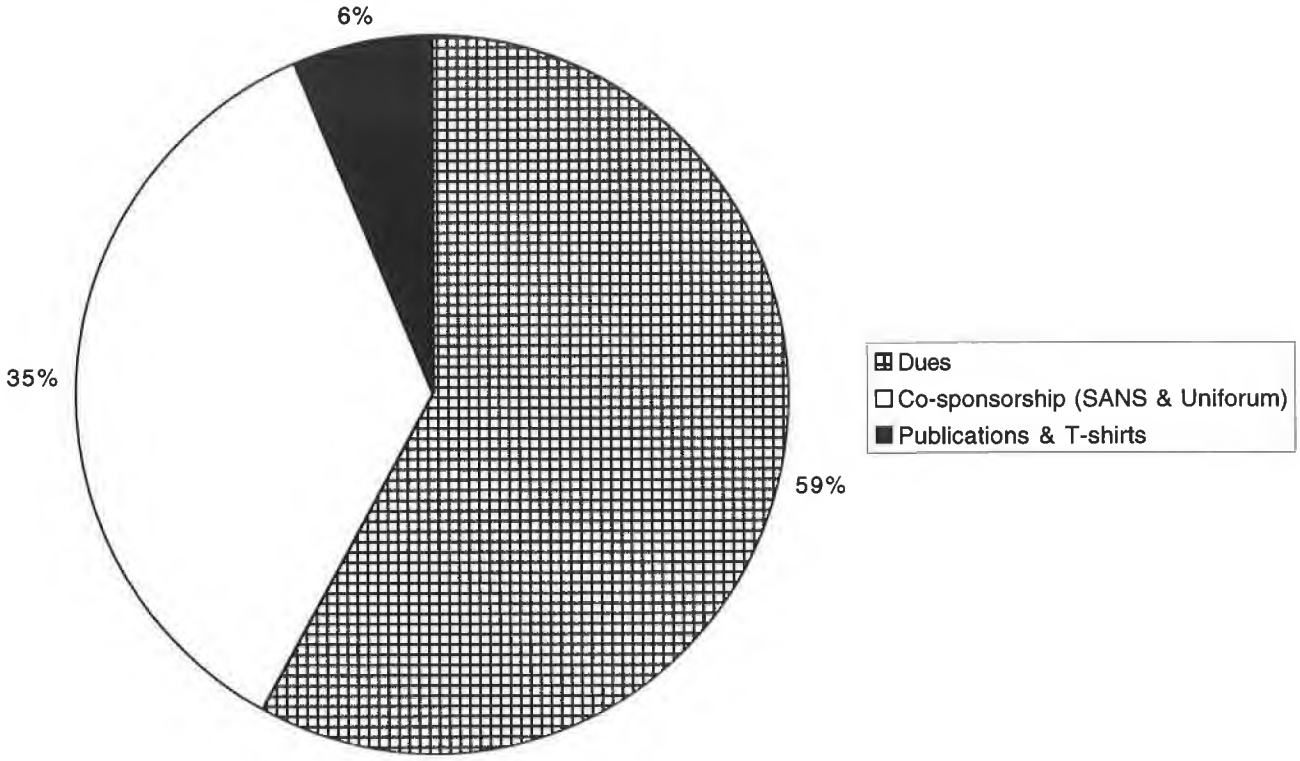
Where Did Your '95 Membership Dues Go?



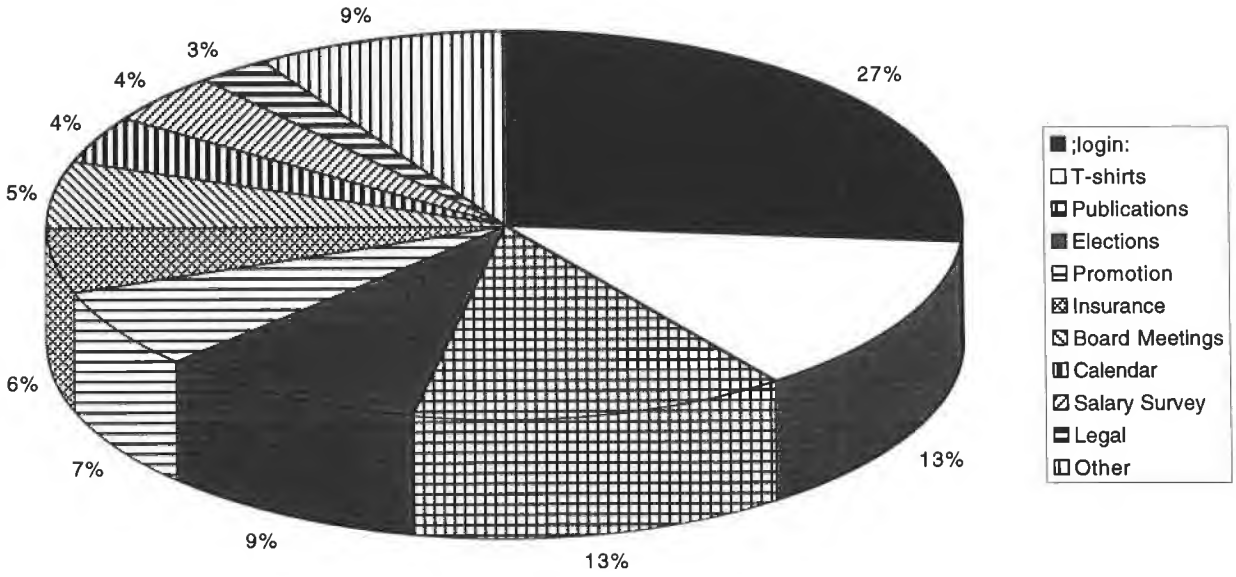
Executive Office Expenses



1995 SAGE Income Sources



1995 SAGE Direct Expenses



USENIX PGP Key-signing Service

by Greg Rose
<Greg_Rose@usenix.org>

Introduction

USENIX is going to provide a new member service: signing members' PGP public keys for the purpose of introducing members to the PGP "Web of Trust" in a simple and convenient manner.

The Web of Trust

There is a publicly, and internationally, available privacy program called PGP. PGP uses public key cryptographic techniques to enable messages to be exchanged between people across public networks while both protecting the privacy of the contents and guaranteeing authenticity of the sender.

Traditionally, one of the problems with cryptographic systems has been "key management." The key is the secret information that enables information to be decoded. Before public key systems came along, the key had to be securely exchanged between the parties before they could communicate. Now, with public key systems, the keys are split into two pieces, one of which can be made public (like a telephone number) and the other of which must be kept secure by the owner (like the telephone itself). Key management would seem to be a solved problem.

Unfortunately, that isn't so. Key management is undeniably easier using public key systems, but the question becomes one of authentication. How do you know, for sure, that the person you think you are sending the secret message to is really the person you wanted to send it to? As an analogy, I could easily get a telephone connected in a false name and sit back waiting for phone calls intended for another person of that name.

One answer to the problem is to have trusted parties who introduce other parties to you. This is what the PGP documentation calls the "Web of Trust." It is a web because each party in it can introduce other parties whom you may or may not already know. Using the telephone analogy, you would say secret things on the phone only if someone you trust had given you the telephone number, not if you had just looked it up in the phone book or heard it from an ad on TV.

Another answer to the problem is to have Certification Authorities that form a hierarchical structure. When you get a public key, you would also get a list of certificates. For example, J. Smith's public key might come with a certificate from Widgets Inc., stating that he works for them. In turn, Widgets Inc., would need a certificate from someone stating that it is a Delaware corporation. The state of Delaware would need a certificate stating that it really was what it said it was, presumably issued by the omnipresent authority (which, at the moment, is RSA Data Security Inc.).

(I intended the above to be the absolute minimum explanation of the concepts of public key cryptography and key management. If the concepts are not yet clear, the PGP documentation, which you should eventually read, explains it in more detail.)

Both schemes have flaws. The big problem with the Web of Trust is that it has to be big and well connected before it is useful, but the Certification Authority model implies a sort of control that is often the reason the parties wanted to communicate privately in the first place.

The USENIX Key-signing Service

Finally, I can tell you what this is all about. USENIX, acting as a party that has no real interest other than the spread of this technology, wants to be able to expand the Web of Trust by acting as an introducer for PGP keys. USENIX staff, by virtue of the conferences they hold, are in a position to meet physically with people, verify their identity, and issue certificates (key signatures) attesting to their identity. In a way, this is like acting as a Certification Authority, but in a situation where this has no hierarchical significance, and there is relatively little left of the "authority." Anyone will easily be able to verify that USENIX's key belongs to USENIX because the key identification information will be published in our newsletters and promotional information; and the organization is well known, so you can always call the office to verify it. In turn, because USENIX's procedures are public, you will know exactly what level of trust to place in the authenticity of another party's public key introduced by USENIX. (We are already talking to other user groups about levels of cross-signing, to form a sort of Web-of-Trust Backbone.)

Our policy about when and how we sign PGP keys, and our procedures for doing so, are described in detail elsewhere. There is really only one thing that is innovative about this.

At the moment, PGP keys are often signed at mutual gatherings (a number of signings have happened at USENIX conferences). Interested parties gather the information needed to verify that a public key is the one they are referring to and print it on a piece of paper. This information is called a *fingerprint.* One of mine, just for an example, is listed at the top of the following page.

At the gathering, I might stand up and say, "I'm Greg Rose, here's my passport and business card, and the key with fin-

Type	bits/keyID	Date	User ID
pub	1024/09D3E64D	1994/11/30	Greg Rose <Greg_Rose@sydney.sterling.com>

Key fingerprint = 35 0A 79 7D 5E 21 8D 47 E3 53 75 66 AC FB D9 45

gerprint 35 0A . . . really does belong to me." The others present can take away the piece of paper and at a later date download the key (or I might send it to them). They verify again that the fingerprint matches and then can sign the key themselves knowing the association between the key and the person who stood up at the meeting.

The big problem with such meetings is that you have to come prepared; this makes it very hard for people who come along to find out what is going on, and haven't prepared the fingerprint (or, indeed, the key itself). Once they've left, the people may have lost their only chance to get connected in the Web of Trust.

Of course, if you do happen to have come prepared, we can handle that. The innovative thing that USENIX offers is that we can reverse that preparation; USENIX comes prepared so you don't have to. USENIX has created some "shared secrets" that we can give to people.

The shared secret is eight four-letter-words, which we pseudorandomly generate based on a secret password. We print the secret on tamperproof media (folded cardboard) prior to use. When you want to make use of the service, you simply come to the USENIX desk at a conference, fill out and sign the appropriate form, produce two forms of identification (one with a photograph) and get one of these secrets from us. You should then keep it safe, because it forms the only link between our identification information and you.

Later, when you get home and have created your PGP key, you send USENIX electronic mail, encrypted to retain the secrecy, and signed by your PGP key to establish that correspondence. Inside the mail, you should include the PGP public key itself, the identifying information from the outside of the secret, and the eight secret words from the inside. We cross-check that the email came from the person we thought it should have and that the secret information is correct. Then (assuming everything is kosher) we sign your key and return it. We will also have a Web page which can be queried for specific keys we have signed and another one that can be checked to be certain we have not revoked our signature on the key. (Note: PGP doesn't support revocation of signatures, so this is the best we can do.)

What is this form we ask you to sign? It is very simple. You state that you are who you say you are and indemnify us from liability if you have misrepresented that. You agree to notify us if you revoke (or lose control of) your own key. Lastly, and this is totally optional, you might want to make payments to USENIX via electronic mail (for things like conference registrations, membership dues, or past proceedings), and authorize us to charge your credit card if you send us the details in signed and encrypted email. You are not committing to pay USENIX anything by doing this, merely opening the path. (USENIX is very explicit about having a signature on paper before accepting credit cards – this authorization is for us.)

Keep an eye on the USENIX Web pages for more details as they become available.

From the Editor of *Computing Systems*

by Dave Presotto
<presotto@plan9.bell-labs.com>

Computing Systems, the USENIX technical journal, is in need of quality submissions. For any journal to remain interesting, it needs to publish material that is both interesting and timely. For this, it requires a large number of submissions from which to choose. That has stopped happening with *CS*. We now have enough submissions to fill the issues, but few more than that. In fact, without invited papers, we might well have skipped one issue. Unless this situation improves, we must either discontinue *CS* or let the quality slip.

Computing Systems was initiated to fill a gap, i.e., a place where systems implementors could publish journal-quality papers. The journal generates no revenue for USENIX. It is meant as a service. We have always promised quick turnaround for reviews (four to eight weeks). We publish quarterly. Accepted papers are normally published within six months of the original submission. This made us fairly unique nearly a decade ago. However, as systems work has become academically reputable, other journals and conference proceedings have become competitors. We now have to decide on the viability of *CS* as a printed journal.

Therefore, I am asking the community to vote with its actions. Send us papers or tell us to close up shop. We are looking for original works about operating systems, tools, language implementations, networks, and environments that you have built, i.e., we tend to the practical. Papers about improvements to such systems are also fair game. Papers about projects that can be evaluated based on some experience are preferred.

Electronic submissions should be sent, preferably in PostScript, to Peter Salus, our Managing Editor, at *<peter@usenix.org>*.
If you have personal comments or suggestions, send them to me at *<presotto@plan9.bell-labs.com>*.

SAGE, the System Administrators Guild, is dedicated to the advancement and recognition of system administration as a profession. In three years, SAGE's membership has increased steadily, and there is growing recognition of SAGE as a representative in system administration issues. SAGE brings together system and network administrators for:

- professional and technical development,
- sharing of problems and solutions,
- communicating with users, management, and vendors on system administration topics.

SAGE News Editor

- Tina Darmohray
  <tmd@usenix.org>

SAGE Board of Directors

- Paul Evans, President
  <ple@usenix.org>
- Tim Gassaway, Secretary
  <gassaway@usenix.org>
- Barb Dijker, Treasurer
  <barb@usenix.org>
- Helen Harrison
  <helen@usenix.org>
- Bryan McDonald
  <bigmac@usenix.org>
- Hal Miller
  <halm@usenix.org>
- Kim Trudel
  <kim@usenix.org>

SAGE Working Groups

Group	Chair
sage-certify	Paul Moriarty
sage-edu	Ron Hall
sage-ethics	Hal Miller
sage-jobs	Tina Darmohray
sage-locals	Rene Gobeyn
sage-online	Pat Wilson
sage-policies	Lee Damon

You can contact these groups via email at *<their-name@usenix.org>* for example, *<sage-certify@usenix.org>*.

SAGE Discussion Groups

sage-security
sage-managers
sage-outreach
sage-pt
sage-solo

SAGE BOFS

sage.board
sage.bof.women

SAGE Online Services

Email server:
majordomo@usenix.org

FTP server:
ftp.sage.usenix.org

WWW URL:
http://www.sage.usenix.org

SAGE Supporting Members

Enterprise Systems Management Corp.
Great Circle Associates
Pencom Systems Inc.

SAGE NEWS

Editorial

by Tina M. Darmohray
SAGE News Editor
<tmd@usenix.org>

We've all witnessed growth of the Internet that exceeded even the most enthusiastic predictions. Still, the claims are for continued exponential growth. Analysts are suggesting that virtually every business, organization, and educational institution will be connected. Electronic commerce is banking on households getting online as well. Does that mean that every house will need a network administrator? I don't think so.

History indicates that for something to become popular with the masses it will have to be really intuitive. Something like the Mac's famous "look and feel," which made using a computer simple enough for anyone to do it. The learning curve can't be too steep, or most folks get frustrated before they get hooked. If every Mac had been complicated enough to require a system administrator, I don't think Macs would have the appeal they have today.

I think getting on the Internet is going to have to be a lot easier than it is now before Joe and Josephine Homeowner will hop on. Things have improved, but there is a long way to go. We're beginning to see packaged software that makes it easy to use a home computer with major online service providers, but I still get a lot of phone calls from desperate people who are grappling with acronyms, attempting to understand multiple options, and overwhelmed with technology. The bottom line is service providers are still requiring too much knowledge from consumers for Internet connections to go mainstream.

Here's what I think needs to happen: an Internet connection is going to have to get as easy for the consumer as a telephone connection is. Today, for residential phone service, the customer contacts a phone company, requests the service, agrees to pay the money, and the installation is scheduled. On the day of the installation, a technician arrives and asks where the phone jack should be located. Usually, within the hour, the phone is plugged in, and the expected dial tone is there. Phone companies don't ask homeowners about line speeds, kinds of software and hardware in their phones, and detailed information for globally registering their phone numbers, naming their prefixes, or switching algorithms to route their calls. Happily, all that complexity is taken care of for the customer. And I think that's where we're headed with Internet connections.

What does this mean for sysadmins who specialize in network connections? I think that the complexity will still exist, but I think that mainstream success will have dictated that it is removed from the end user. Like the technology to run the phone systems, which is impressive, complex, and requires highly experienced individuals to keep it running, the internal workings of the Internet will still require highly skilled system and network administrators. More and more, however, I think the most common Internet connections will be packaged and be straightforward to "install," leaving the place for administrators with the organizations that construct and maintain the complex internets, not with the ordinary connection.

Letter from the SAGE President

by Paul Evans
<ple@usenix.org>

The USENIX Technical Conference in San Diego was the scene for a new burst of SAGE activity, and this column seems a good place to share with SAGE members what's going on. With the new year came change in the composition of the board. Paul Moriarty, Pat Wilson, and Elizabeth Zwicky, all of whom are founders and have been members of the board since the creation of SAGE, retired in January. We thank them for their service, and look forward to their continued contributions to the system administration community in other activities.

We wish to recognize Elizabeth Zwicky in particular for her service as President of SAGE for three of the first four years of its existence. I'm pleased to note that with the recent USENIX election, she will now be serving the Association on its own board of directors. The board was also pleased to welcome our new board members, Barb Dijker, Tim Gassaway, and Helen Harrison. The SAGE Board elected new officers: Barb Dijker is the Treasurer, Tim Gassaway is Secretary, and Paul Evans is President.

At the January board meeting, the board settled on some major areas on which to focus in 1996. It is our goal to publish two to three new pamphlets to enhance the series that started with Tina Darmohray's *Job Description* booklet. We are actively working on a policies booklet, and the board selected from a range of proposals for a *Security for Management* booklet. Other ideas include a pamphlet on legal or ethical issues.

Another goal, which goes along with the first, is to plan our publications for next year. We would like to keep adding each year to a list of high-quality publications for system administrators. There are certainly many opportunities for member involvement in this arena.

The board is eager to enhance SAGE's online presence on the World Wide Web. If you have ideas about what kinds of information and materials SAGE should be making available to its membership in this form, please let us know.

The board is also very concerned with encouraging the formation of new local groups and strengthening existing ones.

Finally, we intend to report to our membership about the status of all publicly stated goals and expectations for the organization. Look forward to more on that topic in future letters in this space.

As always, the SAGE Board of Directors wants to hear from you, our members, about your ideas and concerns. You can contact us at *<sage-board@usenix.org>*.

Elementary Intrusion Detection, Part 2

by Karen Casella
<kcasella@eng.sun.com>

In the last issue of *;login:*, I described one simple method to detect the intrusion of an external penetrator. In this second of two articles on the subject of elementary intrusion detection, I focus on detecting the "insider" attack. This type of intrusion is perpetrated by legitimate system users who attempt to gain privilege or access to restricted resources for which they have not been expressly authorized. Remember, a system that is secured by preventing access from outside may not be safe from inside attacks accomplished by abusive use by authorized users.

To continue the example from the first article, the challenge was to monitor access to a server that contained highly sensitive engineering data. Although most of the user requirements were met by providing elementary boundary access controls and monitoring (as described in the previous article), a few unsolved requirements remained:

1. In the case of multiple failed connection attempts, the sysadmin was to be notified by pager.

2. Only the sysadmin was allowed root access on the system (there had been between four and six people who had the root password). If it was detected that anyone else was able to gain root access, the sys admin was to be paged.

3. If system activity exceeded certain thresholds during certain times of the day, the sysadmin was to be paged. In general, activity on the system was expected to be low and disk space usage was expected to be flat.

4. If system resource utilization changed drastically during certain times of the day, the sysadmin was to be paged.

Most UNIX systems have plenty of tools and utilities available that will help a system administrator to log the information that is needed to detect an internal intruder. The problem is that logging utilities such as the `syslog` daemon generate huge amounts of data about the system, and it is tedious to go through all of it manually. Typically, this information is used to investigate an incident after it has happened rather than to catch someone in the act. What was needed was a way to filter through all the log information automatically and alert a human when certain things occurred. I decided to use the `swatch` [1] program to do this. The `swatch` pro-

gram is designed to monitor system activity by continuously watching a stream input for patterns that are specified in a configuration file.

To simplify things, all logging was consolidated into a single source. The most logical choice for the primary logging utility was `syslog`. Some changes to the `syslog` configuration file and some additional `syslog` input were required to meet all the requirements. The following entries were added or modified in `/etc/syslog.conf`:

```
*.err          /var/log/syslog
auth.notice    /var/log/syslog
user.alert     /var/log/syslog
```

Making these changes would log the information needed for the first two requirements listed above to `/var/log/syslog`. Repeated failed login attempts would be logged by an entry similar to:

```
Apr 4 14:23:47 host1 login: REPEATED LOGIN
    FAILURES ON /dev/pts/10 FROM host2
```

A successful attempt to become root on the system would result in a log entry similar to:

```
Apr 4 14:15:42 host1 su: 'su root' succeeded
    for user1 on /dev/pts/3
```

and a failed attempt:

```
Apr 4 14:25:58 host1 su: 'su root' failed for
    user1 on /dev/pts/3
```

There were no "ready-made" log messages that could be used to satisfy the third and fourth requirements, so I wrote a script that gathered information about system usage and used the `logger(1)` program to send the information to the `syslog` daemon. The script, which was run periodically by `cron`, checked several different system usage parameters against predefined thresholds and against the values that were derived from the previous execution of the script. Not only was the absolute value of any particular usage parameter interesting, but it was important to note any patterns of increased usage. The following is a code snippet from the script that checked just the disk space usage of the root partition and the load average:

```
#!/bin/ksh

#####
# Read previous values
#####

VALS=/usr/local/tmp/.swatchvals
. $VALS

#####
# Root partition filling up
#####
HOUR=`date '+%H'`
MAXROOT=70
MAXDIFF=10
[ $HOUR -gt 18 ] && MAXDIFF=5

let THRESH=$PREVROOT+$MAXDIFF
ROOT=`df -k / | tail -1 | awk '{print $5}' |\
    tr -d '%'`

[ $ROOT -gt $MAXROOT -o $ROOT -gt $THRESH ] && \
    logger -p user.alert -t swatch root
    partition filling up: $ROOT

echo "let PREVROOT=$ROOTQR" > $VALS

#####
# Load average (over last 5 minutes) too high
#####

MAXLOAD=1
MAXDIFF=2
[ $HOUR -gt 18 ] && MAXDIFF=0

let THRESH=$PREVLOAD+$MAXDIFF
LOAD=`uptime | cut -d, -f5`

[ $LOAD -gt $MAXLOAD -o $LOAD -gt $THRESH ]&& \
    logger -p user.alert -t swatch load
    average too high: $LOAD

echo "let PREVLOAD=$LOAD" >> $VALS
```

Now that all information was being logged by the `syslog` daemon, a single configuration file (`.swatchrc`) could be used with the default action of `swatch`, which is using a `tail(1)` of `/var/log/syslog`. In its simplest form, the configuration file for `swatch` is made up of entries containing pairs of patterns and actions. The first field of each entry in the configuration file is a pattern expression that must be regular expressions that `perl` will accept. Each string in the input file is matched against the expressions in the configuration file. If a match is found, the corresponding action that is defined in the second field of each configuration file entry is taken.

The following is an excerpt from the `swatch` configuration file that was used:

```
#
# Swatch configuration file
#
# Multiple failed login attempts:
#
/INVALID|REPEATED/  mail=sysadmin:sapage
#
# Only user "sysadmin" allowed to have root
#access:
#
```

```
/su .* sysadmin/      mail=sysadmin
/su .* succeeded/     mail=sysadmin:sapage
/su .* failed/        mail=sysadmin:sapage
#
# System activity too high or growing too
#fast:
#
/swatch root partition/
                      mail=sysadmin:sapage
/swatch load average/mail=sysadmin:sapage
```

The patterns are simple, as are the actions. For all entries, the action sends the matched line to the system administrator. For those entries that are considered critical, the matched line is also sent to the system administrator's pager.

In this two-article series, I have documented how elementary intrusion detection was applied to one situation that I encountered along the way. There are many other tools that can be used to assist in more complex intrusion detection (`tripwire` and `tiger` come immediately to mind), and there is a great deal of research being done on sophisticated methods for intrusion detection. If you are interested in reading more about intrusion detection, some good Web sites to visit are:

http://www.cs.purdue.edu/coast/coast-tools.html
http://security.cs.ucdavis.edu
http://www.cs.ucsb.edu/~jonwood/ustat.html

A mailing list dedicated to intrusion detection is also available. To subscribe, send mail to *majordomo@uow.edu.au* with the words `SUBSCRIBE ids` in the body of the message.

References:

[1] Stephen E. Hansen, and E. Todd Atkins, "Automated System Monitoring and Notification with Swatch," *LISA IV Proceedings*, November, 1993.

Disclaimer – The information in this article is the opinion of the writer only and does not reflect any particular system monitoring or access control configuration. Sun Microsystems does not endorse this or any particular type of system monitoring or access control configuration.

Perl Practicum: The Devil in the Details

by Hal Pomeranz
<hal@netmarket.com>

A Modest Proposal

We usually think of UNIX tools using command-line switches rather than configuration files. Sometimes, however, the number of configuration options is so large or the amount of configuration information is so daunting (e.g., `sendmail` or `inetd`) that a configuration file is required. Configuration files also give users a relatively easy interface for customization and enable the program to adapt over time without destabilizing the application's actual code.

Lately, I have been writing a lot of applications that cry out for the use of configuration files, and I have found that a particularly flexible paradigm is to write config files in Perl syntax and then `eval()` them. There is no need to write a new file-parsing routine for each application, and the configuration files have direct access to data in the program's environment. The downside, of course, is that it is harder for users (particularly nontechnical users) to configure the application. Still, if you write tools for developers (as I generally do), this is an extremely powerful idea.

Pulling in Files

Generally, there are three ways to bring such configuration files into a program. First, there's the "`open()` the file and slurp" method:

```
open(FILE, $file) ||
    die "Failed to open $file\n";
@lines = <FILE>;
close($file);
eval("@lines");
die "Failed to eval() file $file:\n$@\n"
    if ($@);
```

The variable `$@` is defined only if the preceding `eval()` statement detected a syntax error. Configuration files tend to be small and RAM tends to be large, so there is not much worry in using too much memory for `@lines`. Note that `$file` must be the configuration file's full path name or `$file` must be in the current working directory of the program.

A second option, then, is to use `do $file`. The `do` construct searches the standard Perl "include" path stored in `@INC`. The problem is that `do` won't trap syntax errors in the configuration file. Instead write:

```
$result = do $file;
die "Probable syntax error $file\n" unless
    ($result);
```

and make sure to end `$file` with a nonzero statement as is typical for Perl library files (usually the last line of such files is simply "1;"). If it hits a syntax error, `do` stops evaluating – causing `$result` to be `undef`.

Rather than having to remember to end config files with statements that evaluate to be nonzero, another option is to use `require`, which searches through `@INC` just like `do` but also raises a fatal error if the file contains a syntax error. These errors can be trapped with `eval()` like this:

```
eval('require("$file")');
die "*** Failed to eval() file $file:\n$@\n"
    if ($@);
```

The only difficulty with `require` is that it will include a given file only once. This may not seem like an issue at first, but suppose the application is a daemon that is supposed to re-read its configuration file when it receives a `HUP` signal.

It turns out that `require` keeps track of which files the program has read with the `%INC` hash. The keys to the hash are the arguments given to `require`, and the values are the full pathnames to the file as found by searching `@INC`. Using this information we can write this simple function:

```
sub acquire {
    my($file) = @_;
    delete($INC{$file});
    eval('require("$file")');
    die "*** Failed to eval() file
        $file:\n$@\n" if ($@);
}
```

This gives us all the benefits of `require` and still enables us to reread the same configuration file.

But What Good Is It?

All right, we can safely read in configuration files written as Perl code, but what exactly does this buy us? In one of my early columns, I talked about writing portable Perl scripts that read in a configuration file that contained machine-specific configuration information. For example, consider an `/etc/OSinfo` file on all machines that contained information like:

```
$VENDOR = "Sun";
$HARDWARE = "Sparc";
$OS = "Solaris";
$VERSION = "2.5";
$HOSTNAME = `/usr/bin/uname -n`;
$PSCMD = "/usr/bin/ps -ef";
$MAILER = "/usr/bin/mailx -s";
```

On Berkeley-based systems, `$PSCMD` might be `ps -aux`, and `$HOSTNAME` might be set by calling `hostname` instead of `uname`. All of a site's administration scripts could simply use the `acquire()` function to suck in all this configuration information. Assuming the scripts used the variables set in the configuration file, they would be completely portable across every machine on the network.

Configurations files can, of course, contain more than simple scalar variables. For example, I wrote myself a little program that splits my mailbox up into smaller files based on who the email comes from. The program reads a configuration file which defines a hash like:

```
%File = ("firewalls-owner" => "firewalls",
        "owner-namedroppers" => "dns",
        "bind-" => "dns",
        "socks-owner" => "socks",
        "owner-www-security" => "wwwsec",
        "owner-best-of-security" => "bos",
        "bosslug-owner" => "bosslug",
        "owner-solaris-x86" => "x86",);
```

The keys of the above array are all `From` addresses of various mailing lists I subscribe to. If the `From` address of a given message matches one of the keys in the hash, then the message is deposited in the file whose name is `$File{$key}` (if the `From` address doesn't match any key, then the message goes into a default file). This program is very useful when I've been out of the office for several days and I want to ignore all the mailing list traffic I usually get and just concentrate on mail sent by individuals.

It is even possible to define subroutines in the configuration files. Because the `eval()` statements happen at runtime, function definitions in the config file will always override declarations with the same name in your program. For example, a program like this:

```
eval('require("$file")');
die "*** Failed to eval() file $file:\n$@\n"
        if ($@);

sub printer {
    print "In Perl prog\n";
}

printer();
```

with `$file` that contains this:

```
sub printer {
    print "In required file\n";
}
```

will print `In required file\n` when the configuration file invokes `printer`. My PLOD program uses this idea to enable users to replace a standard (but weak) encryption routine with a stronger routine of their own devising.

Generally avoid using symbols in a configuration file that will clash with variables and function names in the programs. One simple solution is to use all uppercase symbols in configuration files and all lowercase symbols in programs. Symbols in your configuration file could also be prefixed with some standardized string (such as the name of the configuration file itself). Alternatively, use `package` in the configuration file to push all symbols into a protected namespace.

Hybrid Files

Sometimes it is undesirable to force users to write a full-blown Perl script just to configure the latest tool. Instead, consider using a hybrid-type file that has easy-to-parse fields, some of which might be Perl expressions. For example, we could rewrite our `/etc/OSinfo` file as

```
# Sun specific:
VENDOR Sun
HARDWARE Sparc OS Solaris
VERSION 2.5
HOSTNAME `/usr/bin/uname -n`

PSCMD "/usr/bin/ps -ef"
MAILER "/usr/bin/mailx -s"
```

and then read in the file with

```
open(FILE, "/etc/OSinfo") ||
    die "Failed to read config\n";
while (<FILE>) {
    next if (/^(#.*|\s*)$/);
    ($key, $val) = split(/\s+/, $_, 2);
    $Config{$key} = eval($val);
    die "Error on line $.:\n$@\n" if ($@);
}
close(FILE);
```

Note that this code skips comments (lines beginning with a "#") and blank lines (lines that contain only white space). It uses the three-argument form of `split()` so that the line is broken into two pieces. This kind of hybrid file can give the best of both worlds: easy, yet extremely flexible and powerful configuration.

Wrapping Up

Although these kinds of configuration files can be extremely powerful, they can also be a living nightmare for users. Make them have to configure only useful parameters, and don't bury them under a huge number of options. Choose sensible defaults that will work properly in the normal case. Provide users with a variety of well-documented, preconfigured files that they can copy and modify to suit their particular needs.

Modern Name Service Directions

by John Schimmel
<jes@sgi.com>

Long ago, when UNIX was young, all of the configuration and account information for the machine was stored in a small number of local files, such as `/etc/passwd`, and simple routines were written to parse these files into a common structure for C. These routines enabled a programmer to step through the file one line at a time or look up an entry using one part of the line. These routines, `getpwent()`, `getpwnam()`, `getpwuid()`, and their relatives became the standard way to fetch data about users on the system.

When networking was added to UNIX, implementors added a number of new configuration files, such as `/etc/hosts`, and more routines to parse these in a similar manner to those that already existed for the current files.

Universities soon added code to the standard lookup routines for the most used files, like `passwd` and `hosts`, to look in a `dbm` hash file for the information to improve performance on busy systems. After editing the local file, the administrator needed to parse the data into a `dbm` file using a command like `mkhosts`, and the routines in `libc` would read from these if they existed.

For the `hosts` file in particular, trying to maintain a single file containing all the machines that someone may ever want to connect to was virtually impossible, so work on a different system was started. The Internet community eventually standardized on the Domain Name System (DNS), which divides all possible system names into a tree based upon the name and then delegates authority for parts of the tree to different organizations. When a new organization joined the Internet, it was issued a tree of Internet network addresses and a tree of names that it maintained. This meant that no system had to know about the entire namespace, but only about the part of the tree for which it had authority, and how to get to the layer above itself.

To implement DNS, the UNIX programmers simply added code to the standard C library host lookup routines, such as `gethostbyname()`. Since then, other name services have been added to these libraries to extend other services in the same way as DNS extended hosts. By far the most popular was Yellow Pages (YP) by Sun Microsystems (later renamed Network Information Systems [NIS] due to a trademark collision with British Telecom who owned the name Yellow Pages and did not want it used for a SunOS name service).

NIS took the idea of a local `dbm` hash file lookup, used by most large sites at the time, and moved it across the network using its new Remote Procedure Call (RPC) code. Sun's RPC library simply took something that looked like a local procedure call, inlined the arguments into a buffer, sent the buffer over the network to a server, which split the buffer back into the arguments, and called the real procedure. When the procedure was finished the server bundled the output parameters from the function call back up into a buffer and sent it back across the net to the original application. All of this code was stuffed in the C library and into the configuration lookup routines.

NIS was a very simple solution that did not address many of the problems with moving something like configuration file lookups across the net. In particular, NIS and all of the Sun RPC routines were plagued with security problems from the outset. Administrators were already very nervous about having passwords flying around on the network, and having nearly anyone able to request them from a name server just exaggerated the problem. (NIS broadcast binding requests to locate a name server, instead of using a fixed list similar to DNS, and servers are unable to protect against remote systems looking up data that they do not have authority to see.) Simple patch fixes were added for both of these, but they were also easily overcome.

Another growing problem was the continual increase in size and complexity of the name service routines in the C library. With each passing operating system revision came an increasing number of problems located within this code. Code inside of `libc` is very difficult to maintain, because it is the home of the runtime linker and the one library that cannot be overridden with environment variables.

Finally, the database system used by NIS (`dbm`) contained a number of internal limitations that make it impossible to store large maps. `dbm` has a page size of one kilobyte, which limits the total length of a key/data pair in NIS to 1,018 bytes (that is, one kilobyte minus three shorts, used by `dbm`). The maximum number of records that can be placed in an NIS database is about 30,000 due to a problem with the hash algorithm in `dbm`. All numeric strings tend to hash to the same page, so the result is that after around 30,000 records the directory bumps into the 2 GB limit in most UNIX filesystems.

Both NIS and DNS have problems with looking up data by different keys and getting different results. NIS contains map files that store each line in a configuration file by each of the possible keys, and, for maps that key on multiple items in a line, these go into different files such as `hosts.byname` and `hosts.byaddr`. In transferring maps, it was possible to transfer `hosts.byname`, but then run out of space on the filesystem so `hosts.byaddr` failed to be updated. In DNS, addresses act as pointers to names, but a single host can contain multiple names, so looking up a host by address and then by name might result in different records being returned.

A couple of new name servers were released to deal with perceived and actual problems with the existing name services. Hesiod is a name service built on top of DNS by the Project Athena team at the Massachusetts Institute of Technology, and the DCE Cell name services is another RPC - based name service built on top of the new DCE remote procedure call infrastructure.

The standard implementation of DNS found on UNIX machines is the Berkeley Internet Name Daemon (BIND), commonly referred to by the name of the daemon that implements most of the protocol, named. `named` is a persistent daemon that runs on server machines and maintains a local cache of all information looked up through DNS, as well as information about where to go for new information since DNS is widely distributed. Client machines can run `named` as well and get the benefit of having the local cache to draw from. DNS has little more security than NIS.

After Sun released its RPC programming model, the Open Software Foundation began the search for a similar model and decided on the Apollo RPC system. Apollo has since been purchased and consumed by Hewlett Packard. The DCE RPC is very similar in concept to the Sun code, but is far more complex. One thing the DCE model has going for it is that the security model was based on a tested mechanism, Kerberos, a ticket-passing system designed by the Project Athena team at MIT. The Cell Name Server is built on top of this RPC model and thus can use Kerberos security.

Sun's answer to the criticisms of the NIS name server was the introduction of NIS+. NIS+ is a system built on top of a new secure RPC model and using a new database design. The new security is based upon the RSA strong authentication algorithm. RSA is a patented algorithm currently considered one of the strongest known. Its only real drawback is the export restrictions associated with it. Because NIS+ is built on top of secure RPC, the security is usable only within the United States.

Where NIS was a very simple implementation, NIS+ is not. It contains about 20 times as much code and is about as feature rich as any name service has ever been. NIS+ contains a number of new ideas that are very interesting, and some of them are probably worth considering in new designs.

The database for NIS+ is a multikeyed object database that solves the problem with receiving different information

when looking up an element using different keys. The database supports multiple sizes of pages, up to eight kilobytes, so the size limit is somewhat larger than NIS as well. The maximum number of keys that one would want to put into a NIS+ database appears to be in the order of a few thousand, however, because it holds a great deal of information about each object and is relatively inefficient.

NIS+ contains support for hierarchical namespaces in a manner similar to DNS, though the underlying lookup routines and applications have not been extended for this to be useful yet. The concept of logging into a machine at a USENIX conference as *jes@sgi.com* are intriguing in the world view of networking, but it is still pretty unclear whether the NIS+ implementation will ever make this possible.

NIS+ also contains some interesting support of transferring database changes to slave (or replication) name servers. Where NIS needed to copy full maps each time something changed, NIS+ can transfer only the changes. This would be particularly useful if NIS+ were able to support truly large domains.

Finally, NIS+ exhibits a number of advancements in security management. All of the public keys for a domain can be stored in an NIS+ map, and a modified Diffie-Hellman key exchange formula can be used by clients to register keys with the server. Each database row and column can have security limits placed on it so that, in the `passwd` database, for instance, users can update parts of their own record, and none of someone else's.

Recent changes to DNS have been fairly minor in comparison to the massive Sun rewrite because DNS is an Internet standard and is changed only by committee. Support was quietly added for large addresses that will be needed once implementations of IPV6 debut. DNS security has been argued unceasingly for years without getting very far, but implementations of the evolving standard are now starting to show up. And the ability to do dynamic updates of DNS information may actually happen some day soon.

DNS has the same security problems that all name servers have. The most important of these is validation. The client system needs to know that the information it receives back from a query is correct. If the client can be fooled into speaking to a different system, then all sorts of mischief can be accomplished. DNS records have always contained authority information, but the new standard seeks to add an authentication record containing some sort of digital signature sent with the data so the client can be guaranteed that the information is accurate. As in any such system, the difficulty is in managing the keys.

The Dynamic Host Configuration Protocol (DHCP) has been a standard for a while. DHCP supplies client machines with information about themselves so that a systems administrator does not necessarily have to hand configure each new machine that arrives. Its most useful function so far has been in managing part-time connections such as laptop systems. A DHCP server supplies a machine a temporary name and address that it can use for a short time. DNS needs to be changed to support dynamic updates from the DHCP daemon so that other hosts on the network can look up these new names and addresses. The DNS dynamic update protocol still has problems that have stopped it from becoming a standard, such as what to do about security.

Another interesting new standard coming out of the Internet community is the Service Location Protocol. This protocol uses multicast and service registration points similar to the RPC portmapper to locate services across the network. The standard is still incomplete. But it could result in a common binding mechanism for all network services in the future.

Sun Microsystems did something about the increase in code and complexity of the C library by moving the protocol-specific name service code into separate libraries. Now, using the dynamic shared library loader routines, these libraries are automatically brought in when needed. This allows maintainers to test these routines separately and greatly simplifies location of problems. The implementation also supports the addition of new protocols by customers. New name service libraries can be written and added to the name service switch. Many large sites are already using this freedom to their advantage by adding hooks to look into proprietary databases such as Oracle, Sybase, and Informix.

Another aid to the independent developer has been the announcement of the Federated Naming System by the OSF. The Federated Naming System is really just another layer of indirection in the naming code so that all routines call a common lookup procedure. This allows the simple addition of new services to the namespace. In the Sun implementation, each of the library routines such as `getpwnam and getpwuid` need to be implemented, and there is a great deal of replicated code. With the Federated naming system, there are a small number of entry points, and the `libc` naming routines are built on top of these.

All of this new freedom and security in the name services does not come without penalty. The performance of lookups in Solaris has been so bad that with the latest release Sun added a local cache manager to help at least reduce the penalty in repeated lookups. Building an NIS+ database of any size for the first time can take hours because it needs to set up the security objects and sign each record. And the amount of code in modern operating systems has skyrocketed, now exceeding the amount of code needed to implement a filesystem.

Name services are playing an increasingly important role as more and more historically local applications are extending themselves onto the wide open space of the network. Name service development is proceeding toward more secure implementations, supporting larger, more hierarchical namespaces and generalization to allow for faster development of new ideas.

Food for the Line Eater

by Shawn Instenes
<shawni@celene.rain.com>

In my last column, I discussed briefly the IPSEC working group standards (RFCs 1825-1829) and two of the three key management protocols, SKIP and Photuris. The third protocol, called ISAKMP (Internet Security Association and Key Management Protocol), now has prototype code available. See *<http://web.mit.edu/network/isakmp>*, the ISAKMP distribution page. The code has an interesting document that compares the various key management protocols, including two that aren't standards track. There's also a mailing list that discusses ISAKMP, and you can find out more by getting the source.

But I think I've talked enough about the details for a while. I'm going to warm up the crystal ball and talk about how one day we might use these protocols.

Imagine a time when IPv6/IPSEC-capable nodes are "just about" all there is connected to the Net, when an IPv4-only machine is as quaint as "FOOD FOR THE LINE EATER" in USENET postings would be today.

Communication speeds are higher. We have LANs that go a gigabit per second, and even the slow dialup links are multimegabit/second. (Hey, I don't know *how*, the picture is a little snowy on this thing.)

We don't have a firewall, at least as one would exist today. How would it keep up with *all that data*? As always, the routers are shuffling packets as fast as they can, and to make them filter packets out based on anything other than destination would slow our Net connection to barely tolerable speeds of tens of megabits. Instead, we have a group of connection managers (CMs) that make decisions about which computers can talk to what other computers.

This is how my sysadmin-of-the-future explained it to me: A new connection comes into machine A, from machine B. These two machines haven't talked to one another recently, so all of the key-exchange protocol is forwarded by machine A to the nearest available CM. Once the nature of the connection is approved by the CM, a key-generating key is sent to machine A by the manager, so A and B can decide on a communications key and begin talking.

Machine A is allowed to keep the key-generating key for a little while, so the managers don't become too busy. The big advantages of pushing the key management away from the end-node machines to the communications manager are central point of control for all data communication, no dependence on the correctness of any piece of code other than the IPSEC implementations, the key forwarding software, and the communications managers' ability to control what communications take place by identity of end users, rather than by IP number and port, and, finally, speed over solutions where a third-party machine must handle all data traffic.

Back to reality. This vision isn't in the standards or the drafts, but it's a possibility; nothing prevents it other than "A Simple Matter of Code" and a willingness to view these new data security tools as opportunities to rethink how our data are protected and ask, "Does this way still make sense in light of these tools?" If you don't like my vision, you'll be able to use these tools to help create your own, one day.

There's no "silver bullet" and no miracle cure for network security. But these new protocols promise to give us more options with regard to how we can secure data, and that's a start.

Step *Away* from the Computer

Opinion by Barbara L. Dijker
<barb@labyrinth.com>

The power of the machine is intoxicating. Anyone who has been a system administrator for more than three years is clearly under its influence. The system and network present daily challenges and puzzles for us. I think there is a gene for the unrelenting need to prove that the computer can't win. We will make it do our bidding. . . no matter how improbable the request. It *should* work; we'll make it work even if it takes all night! There's just something about a job that needs to be done. And the stack is always there.

For myself, a system administrator for almost ten years, that's what keeps me here. Every day is a new day, a new situation, a new set of problems. Boredom and repetition are not in my dictionary.

Three years ago, someone slapped me in the face and told me to wake up and "step *away* from the computer." It's taken almost that long to actually do it. Oh, I thought I had a "life." But even taking two weeks of vacation a year and having a weekend or weeknight free now and again is not enough. So

I quit my job and flushed my pager down the toilet (figuratively, of course).

Of course I'm not suggesting that you quit your job. For me, that was the only way to reduce my hours. After you've worked 60-80 hours per week for five years, it's hard to convince your boss you need to hire another full-time person when the workload hasn't substantially increased. Your employer will likely expect you to give the job everything you've given it in the past. I have a pattern of leaving jobs and being replaced with two to five people.

Suggestions:

1. If you aren't working more than 50 hours/week, don't do it, period. Avoid any situation where you end up doing the work of more than one person.

2. If you are regularly working more than 50 hours/week, cut back. Cut back now. You can do it gradually, but it must be decisively and significantly. Learn how to vocalize, "no." Learn how to say, "if you pay me time and a half." Discuss your plan with your boss.

And don't go home and surf the net! Replace the hours you were working with introspective time, reading, maybe, or interactive time with friends – not a TV show, actual people. Limit your at-home keyboard time. You'll be surprised at what you find. The world looks a little different without a screen in your face.

But this is all a little easier said than done. I was able to quit my job. For you, more help may be in order. Does this sound like you?

- skipping meals because there's too much work to be done
- sitting at the keyboard for more than two hours without getting up
- lack of sleep or difficulty sleeping
- one or more messages per week you have to let sit overnight before sending (because you were upset)
- urgent pages more than once a week
- working more than 10 hours/day

If so, you have SAD: System Administrator's Disease. The cure is to step *away* from the computer. While you are at the keyboard, you need to follow this one simple rule: keep a one-quart water bottle on your desk, and drink it while you work. Heavy neuron work is very dehydrating. When you get up to relieve yourself, be sure to bring the bottle with you and refill it (with *water,* of course). If you get up to relieve yourself and it's lunch time, then eat lunch. If you get up and it's after 5:00 pm, go home. You see, the trick is to get you *away* from the computer. The rest is easy. Besides, drinking water is very good for you.

The biological urgency is simply a means to force you to do what you may not otherwise be able to do. As you manage to pry yourself from the computer with some regularity, it (and its users) will cease to control you. If you manage to regain control of your own work and life, you can look at things more objectively. You'll have the time and energy to consider your directions and make changes you may have only dreamed about. When I quit my job, I started consulting. I specifically structured my work so clients could *not* reach me outside of business hours (to maintain control). In addition to a flourishing local practice, I've helped start a company that is providing an essential community service. The best bonus is that I was able to focus on improving my health, family, and social life – which had suffered as a result of the stress involved.

An Office MUD for Fun and Profit? Or Maybe Just Better Communication

by Valerie E. Polichar
<valerie@ucsd.edu>

MUDs (Multi-User Domains) have long been used as the basis for multiplayer adventure games and chat environments, but in the last few years they have been explored as general communications environments for as diverse applications as university classrooms and military communications devices. Connecting to and interacting with a MUD server feels very much like playing Zork or adventure with a whole lot of other people logged in and visible as you wander around. You can talk to the folks who are in the same "room" with you ("Now what am I supposed to do with this little bird sitting here singing? Get earplugs?") and "page" those who are far away.

Unlike IRC and multiuser talk, MUDs enable a large range of expression and activity (reading the log of a MUD session is similar to reading a novel). MUD server versatility, with its very configurable privacy options, automated responses, and a sense of location, is the main reason why a MUD is a more useful choice for an office communications environment than IRC, talk, video conferencing, or even the telephone.

Maintaining an office MUD offers a number of advantages for a system administration, systems support, or user support team. Among other things, MUDs offer interactive communi-

cations regardless of location, stress release for your staff, and less painful, more productive staff meetings. They can even offer your customers online support. We've been using a MUD since 1993 for intraoffice communication in the Network Operations User Support office at University of California, San Diego, which serves about 40,000 network users and would-be users.

This article discusses some of the applications, advantages, and shortcomings of an office MUD and touches on how to make the most productive use of the environment. Future articles will cover how to select appropriate server and client code; how to install the code and set up your MUD; service and system administration issues involved in maintaining a MUD; and, for the intrepid, how you might open up your user environment to your customers for online support.

Why Start a MUD?

What's wrong with electronic mail and the telephone? One advantage of MUDs is their ability to combine several types of communication environments.

Communication can occur in realtime, as when you are talking on the phone with someone. It can also include some delay. Sending paper mail back and forth invokes a fairly long delay. Sending electronic mail has a variable delay – the recipient might read it immediately or might be off visiting Australia and not be able to get to it for three weeks. Talking on the phone is immediate (realtime), as long as the other person is there. Every time you need to communicate, it's necessary to make a decision about which one of these is appropriate – if it's 2:00 am, for example, you might send email rather than telephoning or ringing someone's beeper.

Although a MUD doesn't perfectly solve the problem of having to determine the appropriate type of communication, it does ease it by providing multiple types of communication in a single environment. Within the MUD, you can see if someone is online or not and how long s/he has been idle. You may then opt to send MUDmail to someone who is not logged in; page someone who is logged in but idle, knowing s/he will see the message upon returning to the desk (you can even make the person's computer beep in case s/he is napping or working at the desk, or in another window); or page the person to get an immediate response if s/he is around.

In fact, on many MUD servers, options enable you to set up an automatic message to be displayed to anyone trying to page you. You might set up different messages depending on whether you plan to remain online or not. For example, if Cynthia is not on the MUD at the moment, and John and Dan are, but I didn't happen to notice that Cynthia was off-line:

You page Cynthia, JohnB and Dan:
"Do you have time to talk to a customer with a Win95 problem?"
Cynthia is in a meeting right now – free at 11:30!
John's on the phone, dudes!
Dan pages: "Sure, send him/her on in."

So What? I Can Just Walk Down the Hall

Immediacy without having to pester a person face-to-face or on the phone is useful. If you're at work and Bill's in the next office, you don't have to go see if he's on the phone, or phone him and get a busy signal, when all you want to do is ask if he knows where George hid the stapler. You simply ask, and Bill can answer if he's able. Often, he can answer even if he's on the phone!

Valerie shouts: "Okay, who exploded a burrito in the microwave?"
Dan pages: Not me
Cynthia looks at you and laughs: "It wasn't me."
JohnB pages: Sorry, wasn't me
George pages: Uh, oops. I'll clean it up.
Lynn pages: I'm at home. Don't blame me!

I'm at Home. Don't Blame Me!

That last page illustrates a second useful feature of MUDs as a form of intraoffice communication: the ability to communicate equally well from remote locations. My staff comprises ten people; we are currently occupying a space normally allotted to six. As you might imagine, this causes some strain (and probably income for deodorant manufacturers). It's useful to allow some staff members to work from home a day a week, to relieve them and the rest of the office of the strain of overclose proximity. In order that the office not be deprived of their expertise, however, they make PPP or ISDN connections from their home and keep one window on the MUD as they work. This is useful to everyone involved:

Lynn says: "One of our mailing list maintainers is leaving UCSD and wants to continue to maintain the 'bem' mailing list. Should I tell him that he can't, that he needs to find a UCSD person to run it or something?"
Bill says: "I think we should tell him that we cannot maintain it at UCSD any more unless he finds a UCSD user to maintain it, because right now our policy is to allow only UCSD-managed lists."
Dan says: "As long as you're here, Lynn, what should I tell people who come to the door asking about dorm network signups?"
Lynn sighs. "I didn't want to think about them! Have them fill out one of the green forms and leave it in my "in" box. Their hookup will go in within about two weeks, depending on the load in the wiring group. I'll send it in tomorrow"

Lynn kicks the soda machine and twelve sodas fly out! One of the sodas lands in your hand! You now have a can of orange soda pop.

Thanks to the MUD, I've been able to allow three of my four full time staff members to telecommute regularly to some degree.

Incidentally, when staffers are out sick, if they're well enough to log on, they can also keep in touch via the MUD. When I had throat surgery three summers ago, I was confined to bed and couldn't talk without pain for over a month. Thanks to the MUD, though, staff meetings ran on schedule and my staff members were able to ask me questions and get approval or decisions made without having to phone me or go over my head.

You Now Have a Can of Orange Soda Pop

That can of orange soda pop points to a third excellent reason to maintain a MUD for your office: fun and stress relief. Although I can't see starting up a MUD with only this intention in mind (for the same reason I don't distribute game software to all my employees: one does need to work sometimes!), I see it as critical in a high-pressure environment. Users can create their own toys (like the soda machine) and play with them, even in the midst of an otherwise serious conversation. They can also yell things that would be entirely unacceptable to yell in a normal support office!

Dan shouts: "Watch out! Otis is here again!"
Cynthia GROANS!
George throws a pillow at Cynthia.
Peter shouts: "NOT OTIS!!! AAAUUUGGGGHHH!!!"
Cynthia says: "I'd pay him to stay away!"
Valerie says: "C'mon, guys . . . be nice."

As you can see, this can be both desirable (as an outlet) and undesirable (encouraging people to think of their customers in negative terms). For this reason, you may not feel that a MUD is conducive to improving your employees' work. I do believe that having an outlet is critical in high-stress positions such as computer support.

I'm Over at the Medical School . . .

You've probably noticed that, when possible, it's easier to view the output of a crashing program than it is to explain a problem to a customer over the phone. Many of our staff members have to go across campus to help customers get their network software working. They may take a laptop with them when they do so, so they can hook up successfully to our fileserver to download needed files. But if they do that, they can also get onto the MUD, and this means they can tap into the brains of all their on-duty coworkers without having to keep them hanging on a phone line. The staffer can type in, or cut and paste in, the error message and other facts. (The home office could even respond by using the client to quote in the output of an information lookup command, if this would help.)

Dan says: "I'm over at the medical school and I can't get this woman's computer to successfully obtain `bootp`. She's in the Ash building. Is there some problem there?"
JohnB says: "No. Is she getting into the router at all?"
Dan says: "Nope."
JohnB scratches his head: "What kind of Ethernet card is she using? has it worked before?"
Dan says: "error is Bootp Failed: No Dynamic Addressing"
Dan says: "no, it hasn't. Uh, I'll check."
Dan comes back and says, "It's a Putrid Boggleboard."
JohnB looks that up in our database.
JohnB comes back and frowns: "Dan, that isn't working because that board is totally incompatible with the campus LAN. In fact I'd be surprised if it works with anything, given its specs."
Dan says: "She said it was $59.95"
JohnB sighs.
JohnB says: "Break it to her gently."

Not *Another* Meeting!

Staff meetings on the MUD are far less painful than they are in RL (real life). Additionally, staffers are easily able to consult their own personal calendars and look things up in their databases during the meeting itself rather than later. This speeds up decision-making a great deal! In fact, some small tasks can even be taken care of on the spot, simply because every attendee is physically in front of his or her own workstation.

I don't recommend you try to hold any sort of personnel meeting online. Although you can arrange to be fairly sure of privacy, I've found that comments on an employee's work, feedback on interpersonal issues or problems, and other similar issues are best discussed face to face.

This MUD's for You

All in all, the MUD has saved our office time, made home workers more productive, made remote interactions more convenient, and improved morale. The time spent to install and maintain it was more than worthwhile.

In future articles, I'll discuss the nuts and bolts of selecting and setting up your own office MUD. If you are considering a MUD for your office environment but are still unsure, take a look at Rémy Evard's article in the 1993 *LISA Conference Proceedings*, "Collaborative Networked Communication: MUDs as Systems Tools."

The SAGE *Job Descriptions for System Administrators* As A Tool

by Idajean M. Fisher
<ides@psa.pencom.com>

Here's how one company is drawing advantage from the existence of the SAGE booklet *Job Descriptions for System Administrators*. Pencom System Administration is a newly formed business unit of Pencom Systems Incorporated. PSA is made up of and managed entirely by professional system administrators (at present 160+ and growing). We have grown from four members to 160 in a little more than a year and a half. Our business processes are constantly being redesigned to handle the rapid growth. One interesting aspect of this is that the people doing the design and implementation of our infrastructure are PSA system administrators, technical people all.

Many of the PSA team are also members of SAGE. When putting together strategies to manage a large team of people with a wide variety of skills and skill levels, we found that we had to be somewhat creative in our approach to some very basic business practices. We have found the *Job Descriptions* to be a useful framework that can support a variety of standard business requirements. Here are some of the ways we have been able to adapt and use the *Job Descriptions* as a business tool.

Technical Levelling of Potential PSA Candidates

PSA goes through a lengthy technical levelling process for PSA candidates. Because of the complexities involved in establishing well-defined and consistent level determinations, we decided early on to use the *Job Descriptions* to develop working standards. Our current hiring process culminates in a series of technical interviews conducted by PSA team members. By having a person meet with several system administrators from different technical backgrounds with a variety of skill sets, we hope to get a clearer view of the candidate's true skill set and level.

This can be difficult, if not impossible, without standards and commonly accepted reference points. Using arbitrarily selected labels, I may think that a person who has administered 40 to 60 machines for the last five years and knows NIS, NIS+, DNS, and `${Bunch_Of_Other_Stuff}` is a midlevel SA. There may be another member who considers this person to be intermediate and another who believes the person to be junior. Without a standard, all are just arbitrary labels. By familiarizing people involved in the hiring process with the *Job Descriptions* we are giving them a common language set and measurement criteria. If four PSA members talk with a perspective candidate and all understand and employ the *Job Descriptions* as a levelling metric the results suddenly become amazingly consistent.

Employee Review Process

Our employee review process was designed by a working group of PSA team members. They didn't get far into the task before realizing that they had to have objective standards, against which to measure and track technical growth and to help develop realistic and achievable goals. Because we are all system administrators, we have all had the experience of being reviewed by someone who didn't have a clue what we actually do for a living (unless there was smoke pouring out of a system somewhere). We wanted something better than that.

Once again the *Job Descriptions* came into play – this time with some minor modifications. The *Job Descriptions* can be boiled down to a set of checklists with surprisingly little effort. This makes it possible for individuals to take snapshots of their skill sets at regular intervals over time and map progress. It also enables team leaders or PSA mentors to sit down with more junior sysadmins and help them develop a logical plan for growing and augmenting their skill sets with an eye toward helping them move up through the levels.

Training

System administrators need ongoing training for a variety of reasons: to keep current, make better use of their lives, and to stay intellectually stimulated and keep from getting bored. PSA has many system administrators all over the country. Technical interests and needs vary greatly. How do you map, plan for, and assess the success or failure of a particular course of training?

The first level of investigation is obvious. What do people want? However, this really isn't sufficient. It also totally misses the mark when it comes to answering one really basic consideration. What does a particular person *need* to develop his/her career, tie in with an existing skill set, and help him/her progress to the next level as a sysadmin? I might really want to learn high-speed network design, but if I don't already have a basic understanding of standard communications protocols, networking terminology, and concepts, I would probably be wasting my time to jump in and take such a course without starting with the basics.

If you've read this far, you probably know that this is where I'm going to mention the *Job Descriptions*. If a particular member has gone through the review process mentioned earlier and has had the experience of taking a hard look at his/her current inventory of skills, then what s/he *needs* to get to the next level on the curve may become much clearer. Having taken stock of his/her current skill set, s/he can readily sit down with a team leader or PSA mentor and chart a logical course to get to where s/he really wants to be. The answer for our friend interested in high-speed networking might be a basic course on internetworking, followed by a course on standard protocols, and perhaps an assignment that over time will push the person in the direction s/he really wants to go. One of the last steps in the process may be taking a high-level high-speed networking course. Actually, much will probably have to come first.

Interactions with Universities

We are currently working with several universities across the country to set up pilot cooperative education programs. The ultimate goals will be to promote system administration as a career and hopefully help nourish the concept of a degree program in system administration. There is an interesting roadblock to starting this discussion, however. First you have to lay out explicitly what you mean by the term "system administrator." PSA is all over the country. If you are really trying to set up a consistent program, your definitions had better be consistent as well.

In truth, the *Job Descriptions* does not stand up on its own when trying to serve in this capacity. It is a little vague and not easily read by nontechnical people. But when broken down into list form, as mentioned earlier, it can serve as a very valuable starting point. Moreover, you can sit down with people from a university to talk about what type of program you would like to set up, use the interaction to help flesh out the description, then your counterpart in another state can walk into another University and repeat the process. Over time, the list becomes fleshed out in such a way that when a third member goes to speak with people at another university, the task is much, much simpler.

Interactions with Potential Clients

PSA sells system administration expertise. What does that mean? O'Reilly's books could make the same claim, so could the organizers of the next set of LISA tutorials or the people who originally came up with the concept of a man page. If PSA is to be successful, its members must be able to communicate clearly and unambiguously what they can do for a prospective client. Right now we are not truly using the *Job Descriptions* to its full advantage for this purpose. We have standardized the way we describe a PSA member's skill level so that it is consistent with the *Job Descriptions*. This helps more over time as more prospective clients become aware of the *Job Descriptions*' existence. However, the *Job Descriptions* could and should do much more than it currently does to be usable by IS managers and business planners.

Hopes For The Future

There are some modifications that would make the *Job Descriptions* much easier to use as a business tool. At present, the levels need to be adjusted, especially the intermediate/advanced level (it is simply too broad). The range really needs to be broken out into two smaller divisions. The checklist portion of the *Job Descriptions* also needs to be made more concise: a true list format with an eye toward sublevels and shades of gray. Think about someone who overall would be rated an intermediate sysadmin, but in the field of network security could be described as senior. Levelling in reality is field dependent and becoming more so over time as new technologies emerge.

There also has to be room in the description for whatever comes next. System administration is too young as a field to define itself so narrowly. At present, if a particular skill set is not explicitly part of the traditional UNIX sysadmin tool kit, it is not part of the *Job Descriptions*. What about a junior sysadmin who has very strong NT skills or is an expert at running high-speed switch networks? The border conditions also need to be made clearer. Moving to a checklist format would go a long way in this direction, but you will still need to be more quantitative when determining boundary conditions.

Communications and people skills need to be broken out as a separate section. Interpersonal skills, teaching and presentation experience, mentoring experience, demonstrated leadership ability, and other nontechnical considerations should also be broken out and included with their own level descriptions. The ability to make good presentations, mentor other system administrators, and other critical but nontechnical skills need to be considered as well. These skills (possibly more than the technical) are what define us as professionals. They are in many cases measurable. It would even be possible to use snapshots of people's nontechnical skills to try to improve them in a fashion similar to that discussed previously for technical training.

The last consideration is probably the most important. The *Job Descriptions* needs to be understandable by a much wider audience. In a perfect world, those controlling significant computing resources would be technical people with a clear command of their requirements and what skill sets they really need to manage their computing resources efficiently. Some of you may have noticed that this is not a perfect world. Nontechnical people in charge of computing resources need a usable standard much more than we do.

Many businesses that depend heavily on systems people are run by management teams that have only a marginal understanding of good systems concepts, policies, and terminology. Worse, they often suffer under gross misunderstandings of how their systems should work and what they can really do for them. Sadly, when their expectations are not met, this can convert into fear or distrust of the people who set up, design, and administer their systems. In other words, people like *us*! We've all heard an exasperated senior manager say disparaging things that start with "But the marketing people said" or "What do you mean I need more than one administrator to go live 24 X 7?"

Any standard we can hand nontechnical management teams out there in industry that can improve their understanding of what we really do for a living will go a significant distance towards improving the lives of professional system administrators.

USELINUX Conference

by Jon "maddog" Hall
<maddog@usenix.org>

USENIX and Linux International (a fledgling, international, volunteer organization dedicated to promoting Linux) have banded together to sponsor the first Linux Applications Development and Deployment Conference (USELINUX). It will be held in conjunction with the USENIX Annual Technical Conference in Anaheim, CA, January 6-10, 1997.

Recognizing that the two key issues around Linux are the lack of commercial applications and the misunderstanding of the Linux market, USELINUX will present tutorials and three days of technical and business sessions at USENIX. These will include:

- technical sessions for developers of the Linux operating system
- technical sessions for application developers, systems integrators, and systems administrators
- business sessions for people wishing to learn now to increase their market revenue by utilizing the Linux operating system

Business professionals (those in suits) will come to learn how the Linux operating system can be used to increase their business, for it is in this way that Linux will move out of the closet and into its rightful place in the computer marketplace. However, the technical and business sessions will be well defined, and no "techie" will have to be exposed to drivel about how to make money with Linux . . . unless they want to be.

Two committees have been formed, one for the technical program and one for the business program (I am chairing the business track). See page 58 for the announcement.

We urge you to support this effort with your ideas, input, and also by helping us publicize this event to your favorite application vendors.

Interview with Jim Waldo

by Rob Kolstad
<kolstad@BSDI.COM>

Jim Waldo got into programming when he found that his Ph.D. in philosophy did not guarantee him a job in Academia. After paying his dues writing custom business applications in COBOL, he started a long journey doing object-oriented programming of one kind or another on a variety of workstation platforms. While at Apollo Computer, he became an early adopter of C++, used that language to implement an early version of what has become CORBA, and gave up on the language after moving to Sun Microsystems Laboratories in 1992.

While at Sun Labs, Jim has led a group researching how to program large distributed systems. About two years ago he started using an experimental language then known as Oak. The language is now known as Java, and Jim and his group are about to be moved out of Sun Labs to do work in the new JavaSoft organization in Sun.

Rob: What's the big picture on Java?

Jim: The big picture of Java is probably way too big – all of the hype surrounding it has blown expectations out of proportion. That picture has Java changing the way the laws of physics work, unseating the dark empire, and generally making the world a better place for democracy, the American way of life, and the software industry. I wouldn't mind if it did all of these things, but I don't think I want to predict that it will.

I can tell you what I think is good about Java. First of all, it is important to remember that Java is really two things – an object-oriented programming language with a compiler that produces bytecodes, and the environment in which those bytecodes get run. These are distinct entities. There is no reason that someone couldn't produce a compiler for Java that produced machine code, and there have already been projects that compile other languages into a form that can be used by the Java runtime environment (a.k.a. the Java Virtual Machine).

I like the language and find it a comfortable one in which to work. It doesn't have any new features, but uses a lot of the best features from other languages. It looks like a vastly simplified version of C++, but has a distinct notion of interfaces like Objective C. The class structure is like that found in Smalltalk. It has a package notion somewhat like the Modula-3 module construct. In fact, I often describe the language as Modula-3 with a politically correct syntax.

I wouldn't want to argue that it is the best language around, mostly because I've gotten too old to get into arguments like that. I think that it is easy to show that some languages are too complex (PL/1 and C++ spring to mind), but arguing that one language is better than another, after you get beyond the obvious losers, is not something I find very interesting.

What I find a lot more interesting is the Java runtime environment, or the Virtual Machine. This is the interpreter and built-in libraries that try to make all machines look the same to a Java program. This is the part of the overall system that lets Java binaries (the bytecodes generated by the compiler) run on any machine of any architecture. Portability in the Java world means that the binaries are portable, not that you have a reasonable chance of recompiling the source and having it run.

[*Editor's Note: Jim Waldo* <waldo@East.Sun.COM> *will be teaching a tutorial on Java at the upcoming USENIX Conference on Object-Oriented Technologies and Systems (COOTS) in June. See page 60 for full program.*]

Rob: How does Java differ from Perl5?

Jim: Let me begin by admitting that I don't know Perl well. I don't write that many scripts, and when I do, my scripting language of choice is the Bourne shell, with occasional use of Tcl.

Given what I do know about Perl, though, there are a lot of surface similarities. Both are interpreted, both have a certain degree of portability, and a lot of people say that you can use them for the same things.

However, I think of them as very different kinds of languages, used for very different kinds of things. Scripting languages are for rapid development of fairly simple things, or quick one-of-a-kind programs that aren't going to have to be maintained or modified. Perl is, from all I've heard, a great scripting language.

Java, on the other hand, isn't a scripting language. It's a programming language, more on a par with C, C++, or Modula-3. It has lots of nice type checking, garbage collection, and lots of other features that help programmers build really large systems that have to be maintained, changed, and enhanced over time.

So I really think that Java and Perl are different sorts of beasts. There will always be people who use Perl and other scripting languages to do large-scale programming, and there will always be people that use Java for scripting. But I think there is a difference in the two sorts of tools, and they form distinct kinds of entities.

Rob: It seems like a lot of Java's security depends on its inability to get to data in the file system and also system calls. Nevertheless, I/O seems like such a good idea for computers. What's really going on?

Jim: There is so much going on in this question, it is hard to know where to begin.

First of all, the restriction on I/O is not intrinsic to Java, but rather a policy that is established by most of the browsers used to view Java applets. If you are writing Java applications, there need be no such restriction. You could also have browsers that have fewer restrictions. However, given the current distrust of programs loaded over the Internet, it is probably a reasonable policy to exclude any I/O. But again, this is for applets that are being brought in from random locations. If you are writing an application, or have some company internal network where you can trust the code that is being shipped around, you don't need to have these sorts of restrictions.

Next, there is a lot of safety that is offered by Java that has nothing to do with restrictions on what files can be read or written. Notice that I said safety rather than security. The Java goal is to keep your computing environment from harm inflicted, either knowingly or unknowingly, by programs that you download and that get run in the Java virtual machine. This is a kind of security, but doesn't include other notions like access control, authentication, or privacy, all of which also have to do with security.

The safety features of the language have to do with not being able to play around with pointers (Java doesn't have them, it uses real references), not being able to make unsafe type casts, and not being able to mess around outside the bounds of an array. All these features were introduced to help avoid common programming errors. But these are also the way that many viruses gain access to parts of the machine that they shouldn't be touching. The Java language doesn't allow this sort of playing around with the contents of memory, and the loader checks the bytecodes to make sure that the rules haven't been violated in the binary. What goes on is a lot more complex than I can go into here, but it is a pretty neat area of work. We don't have it perfect, yet (as the guys at Princeton keep reminding us), but so far the holes that have turned up have been bugs in the implementation, not problems in the overall approach.

I find it interesting that, on one side, Java is criticized for not having enough functionality, while on the other it is criticized for not offering enough security. People need to realize that they can't get complete safety and general functionality. There is always a trade-off.

Rob: How come people complain about Java's speed? It seems like interpreters can run pretty quick on today's machines.

Jim: A lot of interpreters run pretty quick because they don't do much other than call routines that have been written in C and compiled for the machine on which the interpreter is running. This makes the interpreted language run fast, but also makes the environment pretty large.

There is an active attempt to keep the Java VM small, and to write as much of the environment in Java as possible. This means that instead of calling a lot of pre-compiled C code, the computationally intensive stuff done in Java is still done in the interpreter.

It's still really fast for an interpreted language, and the speed depends a lot on what is being done. On pure computational stuff (for example, standard benchmarks) it tends to look pretty bad. In real life stuff, it does better – we built an RPC system much like the Modula-3 Network Objects system, and found that it ran as fast or a little faster. And M3 is a compiled language.

This doesn't mean that it is as fast as C. But it's pretty fast, and getting better.

Rob: So we can expect performance increases in the near future?

Jim: I think so. There is a lot of speed that can be gained just by making the current interpreter better. There is also a lot of speed that will be gained by what are called "just-in-time" compilers. These are compilers that will take Java bytecodes and compile them down to native machine code just before the first execution of the class.

These compilers are sort of interesting in themselves, because they aren't like normal compilers. A normal compiler could be built for the Java language, of course, but then you would lose the advantage of having portable binaries. These JIT compilers are more like loaders, that also transform the intermediate form bytecodes into machine code. This keeps the portability but enables increased execution speed.

But this makes the compilers sort of interesting. Instead of being run once during development, they are run all the time. So these compilers have to be fast, and not take up much memory, and trade off the high optimization for a small footprint and compilation speed. This is a different set of rules than most UNIX compiler writers are used to.

Rob: There's talk of a Java machine. What does that mean? Will it have an operating system? How will it differ from a general purpose machine?

Jim: There are a couple of things that could be meant by this. The first is the so called "Java Chip," which would run the Java bytecode instruction set as its native instruction set. I don't know enough about how to design a chip's instruction set to know if this is a good idea, but it sounds neat.

The second is a machine that runs the Java VM on bare metal, no matter what that bare metal is. This would be the so called "network appliance" that would only run a browser and Java applications. It wouldn't have an operating system in the usual sense of the term, since it wouldn't have a file system, wouldn't be multi-process, wouldn't have a device driver architecture, etc.

Some have called this a return to a terminal, but it could be a terminal in which most of the processing was done on the terminal by any of the powerful chips that are around today. The hope is that this would give the information sharing and ease of administration that we used to have with departmental minis and corporate mainframes with the local processing power that made workstations and PCs so popular. I think we'll see these machines pretty soon.

Rob: When there was one UNIX, everyone was bug-for-bug compatible. Likewise, when there was one C compiler (or one BASIC interpreter, etc.), there was relatively perfect compatibility on all systems. So far, this is still true for Java and Perl. When there are multiple vendors, regrettably, this compatibility seems to fly out the window (look at all the different HTML extensions out there, for example). Will this happen to Java?

Jim: This is probably the single greatest concern for those of us doing Java. The strength of Java is the homogeneous platform it provides for doing computing on internet-size distributed systems. But this only works if Java stays the same everywhere. Certainly this is what we worry about most when negotiating a license for Java.

But contracts and lawyers aren't the way to insure compatibility. We have to hope that people realize that it is to everyone's advantage to keep Java implementations the same, down to bug-for-bug compatibility. If people start thinking that there is advantage to be gained by having a "better" Java, we will all lose, since much of the real power comes from the portable platform that a standard Java provides.

Rob: It has been said that Java is a dream-come-true for those with Computer Science Master's degrees. Do you think that only super-skilled programmers can use Java?

Jim: I don't think that you need to be super-skilled to use Java, but as I said before when talking about Java and Perl, Java is a real programming language. Programming is hard, and not everyone can do it. Further, Java is an object-oriented programming language, and not everyone gets the Zen of object orientation.

But I don't think you need a Master's degree in computer science to program in Java. I know of a couple of schools that are using Java as the language in the first or second undergraduate programming course. It's the language I will be teaching my kids when they want to learn how to program. I think it is far easier than C++ to learn, read, and understand, and there are plenty of C++ programmers out there.

Here we get to the crux of why I like the Java programming language. It really is, at base, a simple language. There is no pointer arithmetic. There is no operator overloading. You don't have to deal with all of the various complexities of memory management. There is even a pretty good set of libraries that give you most of the basic data structures that you need.

So I find that my Java programs tend to be short (a couple hundred lines), that they tend to concentrate on the problem at hand (rather than building the infrastructure to deal with the problem), and use a lot of preexisting code. I think that makes programming in Java easier than programming in most other languages. I like a language in which I write less code, and in which the code tends to work the first time it compiles. Java gives me that sort of environment.

The Webmaster

by Dave Taylor
<taylor@intuitive.com>

Random Graphics with a Shell Script

Last issue we spent some time looking at how to analyze log files and how the power and capabilities of UNIX make it easy and fun to do so. Well, maybe not fun. In any case, prior to that, I've been bringing you along on somewhat of a guided tour of Common Gateway Interface (CGI) programming, introducing you to the mechanism whereby information is sent between the browser and the Web server.

This time I'd like to continue by developing a short shell script that I think will get your creative juices flowing, a shell script that lets you randomly pick one of a set of graphics on the fly, everywhere in your HTML files that you currently specify a single image.

Content-type

The key to accomplishing this is to learn that the information sent from the server to the client is packaged in a wrapper that specifies the type of information and how the browser should interpret it. In earlier scripts we've slavishly included `Content-type: text/html` as the header to our CGI output. Add a blank line, and anything subsequent is actually HTML that we could easily be generating on the fly. An interesting example:

```
# who is logged in right now
echo "Content-type: text/html"
echo ""
echo "<html><body><pre>"
    who
echo "</pre></body></html">
exit 0
```

Instead of sending HTML, however, what if you could send along a graphic? Obviously, the reference for the browser would be different, but the concept is remarkably similar. The header in question changes to `Content-type: image/gif`, and our script, which we'll call `happy.cgi`, looks like this:

```
# show us the default graphic
echo "Content-type: image/gif"
echo ""
cat happy.gif
exit 0
```

Within our HTML code, we could now include this CGI script as an image directly, as in:

```
<BODY>
<CENTER>
<IMG SRC="happy.cgi">
</CENTER>
```

Random Graphics

A small amount of shell scripting will extend this graphics display CGI script to enable us to choose between a set of graphics in the specified directory. There are two important pieces of this puzzle: obtaining the requested directory from the Web client and randomly picking one of a collection of files from a list.

Sending along an argument to the shell script is easy; any URL can have additional information appended by using the `?arg` notation. Anything after the question mark character itself is handed to the CGI script as environment variable named `QUERY_STRING`. For example, an HTML snippet might be:

```
<BODY>
<CENTER>
<IMG SRC="happy.cgi?images/banner">
</CENTER>
```

The contents of `QUERY_STRING` would be `images/banner`, and the CGI program `happy.cgi` would receive that value.

Randoline

Randomly picking one of a set of elements is a bit more tricky in UNIX, but some variants of the system have a rather peculiar command `random` that picks one or more lines from a file. The Korn shell also has a built-in `random` function, and you can certainly write your own `randoline` program with a dozen or so lines of C. Here's a simple C program I've written to accomplish this useful task:

```
#include <stdio.h>
main(int argc, char **argv)
{
    int total_lines;
    if (argc != 2)
        exit(fprintf(stderr, "Usage:
            randoline lines-in-file\n",
            argc));
    total_lines = atoi(argv[1]);
    pick_line(abs(((int) random() * 32767) %
            total_lines),
        total_lines);
}

pick_line(int myline, total_lines)
{
    int lines = 0;
    char buffer[512];
    while (lines++ < myline) gets
```

```
            (buffer, 512);
    printf("%s", buffer);
}
```

For the random program to work correctly, tell it the total number of lines in the file. To obtain a single randomly chosen line from a file containing 36 lines of information, you'd use:

```
randoline 36 < sourcefile
```

Of course, if you're a Perl hacker, you already know how to accomplish this in a succinct (if unreadable) fashion:

```
#!/usr/bin/perl
srand; @in = <>;
print @in[rand($#in)];
```

Now we have all the tools we need to build our random GIF shell script. Let's look at it in blocks:

```
# rand-graph.cgi - Randomly pick a graphic
# from a set of graphics # in a folder and feed
# it back to a Web page as a GIF image

webhome="/home/taylor"
directory_not_found="/home/taylor
          /dir-not-found.gif"
no_gifs_in_directory="/home/taylor
          /no-GIF-files-in-dir.gif"
```

The graphics files with error messages are required in case the script encounters an error. It can't output to standard error, but we certainly don't want to just have a blank spot instead of our expected graphic.

The first piece of this `sh` shell script looks to see if there's a directory specified in `QUERY_STRING`, and whether it exists within the `$webhome` space on the file system:

```
if [ "$QUERY_STRING" != "" ] ; then
    if [ ! -d $webhome/$QUERY_STRING ] ; then
        file=$directory_not_found
    else
        cd $webhome/$QUERY_STRING
```

Having accomplished this test, we need to ascertain how many GIF files are present in the directory. Rather than using `ls *.gif` however, we'll use `ls | grep .gif` because the former will fail with an error message if there are no matching files, but the latter simply generates a null result, which is much easier to catch:

```
filecount="`ls | grep \\.gif\$ | wc -l`"

if [ $filecount -eq 0 ] ; then
    file=$no_gifs_in_directory
else
```

Now the fun part. We know how many graphics files are in the directory, so we again use `ls` to list them, and stick randoline into the stream to pick one of these entries randomly, saving the result as the variable `$file`:

```
        file="`ls * | grep .gif | randoline
            $filecount `"
    fi
fi
```

If we don't specify a directory with `QUERY_STRING` in the URL, the alternative is to scan the home directory and produce a random graphic from the set of graphics therein:

```
else
    cd $webhome
    filecount="`ls *.gif | wc -l`"
    if [ $filecount -eq 0 ] ; then
        file=$no_gifs_in_directory
    else
        file="`ls * | grep .gif | randoline
            $filecount`""
    fi
fi
```

Having done all this, we've accomplished our goal; the `$file` variable is the name of a file that contains either a randomly chosen graphic or one of our graphic error messages. Now all that's left is to output the appropriate information to the browser and the contents of the graphic:

```
echo "Content-type: image/gif"
echo ""
cat $file

exit 0
```

That's the entire script. With this script and the randoline program installed on your system, you can easily have a rotating list of "cool graphics" such that each time someone loads your page the graphical elements are chosen randomly. If they reload the page, the graphic changes like magic, and the HTML underneath is very simple.

An on-line copy of this shell script and a slightly more sophisticated randoline (along with all my previous columns and lots of other fun and useful stuff) can be found at *http://www.intuitive.com/*.

Letter to the Editor

by Billy Barron
<billy@utdallas.edu>

I'd like to make an addition to Dave Taylor's "The Webmaster: Traffic Jam on Iway-80" (published in the April 1996 issue of *;login:*). Everything stated is correct about the Agent log and MS Internet Explorer. However, one useful fact is

missing. To find many of those IEs pretending to be Mozilla, look for the string "MSIE" in the Agent Log. An example of this appears in the article on the line "151 compatible; MSIE 2.0; Windows 95)." which is an example of Internet Explorer running on Windows 95 pretending to be Mozilla.

Musings

by Rik Farrow
<rik@spirit.com>

Perhaps I'd better start calling it ravings instead of musings. Of course, we all live in exciting times (an ancient Chinese curse – think about it), and I have no special reason for complaining. Well, just maybe I do.

You see, late last year I became interested in Java. Not having a nifty new SPARCstation running Solaris 2 lying around, I found myself faced with installing and using Windows, in particular, Windows 95. I personally found myself uncomfortable with the notion of paying money for an unpleasant operating system, but didn't see any way around it.

Actually, Windows was not my first venture into the non-UNIX, little computer world. I finally caved in and bought a Mac so my wife could help support my home office. My thinking went: my wife is an artist, and the Mac is definitely the artists' computer. Having worked with the Mac a bit, I decided maybe it is a lot like some artists – rather quirky and not totally reliable. (Note that my wife does *not* fall into this category.)

Not being able to type ahead is annoying, but not nearly as annoying as having a 100 MHz RISC processor saddled with an operating system that has a hard time printing while doing anything else. Or worse, just stick a floppy disk into the slot and try formatting it. Now you are stuck for the duration, doing a trivial, background task. At least the Mac recognizes that you *have* inserted a floppy disk, unlike its competition, Windows.

Of everything that has been hard to get used to, the unreliability of the operating system has been the worst. For a while, system crashes (where the cute little bomb icon appears) were annoyingly common. After some crashes, even the power switch no longer worked (in this case, you pull the plug, wait a few seconds, and try again). A partial upgrade of some system software (a very mysterious process, by the way) cured most of this.

But not all. I have a local Internet account for bulk file transfers and often use the Mac for this purpose or just for Web crawling. The Mac's TCP/IP software works okay for the first PPP connection, but the Mac must be rebooted before you can make another connection, use the local Ethernet, or make any configuration changes. The Mac faithful tell me to Wait, Things Will Get Better With Copeland. I sure hope so, but am not holding my breath.

Hell

My daughter, who now works for an "Internet company," recently sent me a cute joke about Bill Gates and what happens after he dies. The short version goes like this. Gates meets St. Peter, who tells him that his is an unusual case, and Gates gets to decide whether he would prefer heaven or hell. Gates, being a canny operator, asks to see his choices first. Hell is a tropical beach complete with bikini clad young women; heaven looks quite boring. Gates chooses hell and vanishes. Two weeks later, St. Peter looks in upon his new guest. Gates, chained to a wall and tortured by daemons, says to St. Peter, "This isn't what you showed me." St. Peter answers, "No, that was the demo version."

I've thought about this again and again as I blunder my way through Windows. Because I want to use the Java Developer's Kit, I must create files with long names, for example, HelloWorld.java. Windows 95 and NT do support long filenames, but maintain a DOS world illusion within the MS/DOS window used for the command line interface. This is unpleasant, but I quickly decided to use the GUI interface to name my files. Oh, oh. Windows 95 is too clever for me, and adds its own suffixes to the files I create. This wouldn't be so bad if the extensions were actually displayed so I could see what has happened. Instead, the file-typing extensions are invisible by default.

When I finally managed to create a text file with the `.java` extension (and nothing else), it came time to compile it. Entering `javac HelloWorld.java` invokes the compiler, which causes Windows to pop up a special window for this windowless application. `Javac` grinds through its initialization, and if there are any problems, these appear for about 150 milliseconds before the window closes. Yes, I have set the Preferences correctly, but to no avail. The easy answer was to create a `.bat` file (poor person's shell script), which means a new window is not opened.

Windows 95 is slightly better than the Mac when it comes to making PPP links. You can do it more than once without rebooting (wow), but configuration is still somewhat mysterious. In trying to get the chat script for login negotiations working correctly, I selected an option that pops up a window so I can watch what is happening. Now I can't get rid of this window and have taken to logging in manually. When I return control to Windows, it sometimes prompts me for a Windows password (something I had wanted to add but wonderful Windows wasn't letting me). I thought I had finally added a login password, but alas, Windows was just faking me out.

Reliable

So I find myself in a maze with lots of twisty, windy passages, all looking exactly alike. My nice new notebook, which will be running Linux really soon, doesn't behave reliably, like a good Turing machine should. Artificial intelligence? No, but perhaps some supernatural agency has taken control of my notebook.

Speaking of Linux, I was sent a copy of the *Running Linux Companion Guide*, complete with CD-ROMs, created by a joint adventure of Red Hat Software and O'Reilly and Associates. Sitting at the end of a skinny Internet link, I really like the idea of getting Linux off of CD-ROMs instead of from sunsite. The slim accompanying book can also be acquired from the Net, but it is nice to have someone else do the job of typesetting. The book is an installation guide so you have something besides the README files, or what's in the `/usr/doc/HowTo` directory to get you started. I may have more to say about Linux next time, when I will be writing this in the comfort of my home instead of cramped inside a plane heading into a snowstorm (and it was 88 degrees when I left Arizona).

Speaking of books, Laura Lemay and Charles Perkins's *Teach Yourself Java in 21 Days* has turned out to be a winner. Hal Stern had told me this would be a good book. I've been through two other Java books, and this is the most thorough by far. There are pages of typos posted to a Web site, so you can test yourself as you go by trying to pick out the mistakes in the code (those in the text are easy). Notice that I don't consider these typos a terrible problem. They should go away in the next printing, and the book is worth some inconvenience.

O'Reilly's *Java Nutshell* (by David Flanagan) looks good, too. More of a reference work (like the other Nutshell Handbooks), it does provide you with enough programming to get started. But the real value lies in the class references and the seven different API cross-references. Something I wanted to see (but didn't) was page numbers in the API cross-references pointing to where the class is described in the chapters on Packages.

Ravings

I haven't heard much about my ravings from the April issue of *;login:*. Having delved deeper into Java, I still think it, or something like it, will have a great influence in the near future of computing.

My rantings about DCE in the February issue got a total of three responses – two from the vendors (each of whom was mentioned in that column) and one from a programmer who supports DCE for IBM in Austin. Perhaps berating DCE to USENIX members isn't fair, but I had hoped for more response than I got.

By the time you get this issue, the Security Symposium will be about a month away. Perhaps at that time I will get what I am looking for, but perhaps you can help me. I am looking for good tools for maintaining network security. I've heard about Argus, a tool that collects Tcpdump-style packet information and transforms it into transaction records, but am looking for more concrete examples about how it can be used for intrusion detection. (For a Web site with a good list of UNIX security tools, try *http://www.alw.nih.gov/ Security/ prog-full.html.)*

And I am always looking for tools that make the process of administering to a distributed, heterogeneous network of computers not only easier, but reasonable. The latest CERT Summary points out that breakins of UNIX systems follow predictable patterns – vulnerabilities in unpatched versions of UNIX systems are often exploited. Even worse, lots of the problems were reported by CERT years ago. We all know how hard it is to keep every system correctly updated, and I yearn for an excellent solution to this rather intractable problem. Maybe in that stack of papers submitted for the symposium there'll be an answer.

Early Insecurity

Peter H. Salus
<peter@usenix.org>

[*An earlier version of this article was given as the Keynote Address at Network Security in Washington, DC, on November 16, 1995.*]

Prehistory

By and large, in the 1950s and the early 1960s, you needed a white lab coat and a badge to gain access to the behemoths we thought of as computers. These giants were fed thousands of watts and more thousands of punch cards and emitted megacalories and either more cards or reams of accordion-pleated paper.

The computer security that was considered was one that involved sabotage or actual theft of paper.

The multiuser machine gave rise to one set of problems. Perhaps the most famous incident of the early 1960s was when a systems administrator on the CTSS machine (at MIT) was editing the password file and another systems administrator was editing the MOTD. Due to a "software design error," the temporary editor files of the two were interchanged and the entire password file was printed on every terminal at login.

The MULTICS project at MIT, an ARPA-funded consortium of AT&T, Honeywell, GE, and MIT, brought us into the world of a modular system with many student users, multiuser capability (at least in theory), and a goal of security levels. This last was for military purposes: MULTICS machines were intended to be both resistant to external attack and to protect users' data from other users. Information was marked "Unclassified," "Confidential," "Secret," and "Top Secret," and could coexist on the same machine, where the OS would ensure that information wouldn't find its way to the wrong user. Eventually, long after AT&T had withdrawn and GE had sold its computing business to Honeywell, MULTICS supplied a level of both security and service most of our current systems still haven't matched.

But in the spring of 1969, MULTICS was far behind schedule and AT&T pulled out. Without going into detail, the net result was that Ken Thompson, Dennis Ritchie, Rudd Canaday, and Doug McIlroy created UNIX in the summer and autumn of 1969. The popularity of the new OS was such that within a few years there were hundreds of sites using UNIX in over a dozen countries.

At virtually the same time, ARPA was funding an experiment in networking. In December 1968, it contracted with BBN to build a packet-switching network of four sites by the end of 1969. The first Interface Message Processor went into place at UCLA on September 2, 1969. The second went to SRI in Menlo Park in October; the third to UCSB in November, and the fourth to the University of Utah in December. The IMPs – which were built around Honeywell 516s – were linked by dedicated 55 kbps phone lines.

There were two programs (what we would now call protocols): Telnet and FTP.

UNIX 1970-1974

MULTICS had passwords as well as security levels; UNIX had no passwords and little else. Why should it? Certainly, on the PDP-7 Ken wasn't concerned with what Dennis might do. Even when the PDP-11/20 arrived in the summer of 1970, no one worried. The machine (and its various upgrades to other PDP-11s) was used for two purposes: development and text processing.

First Edition UNIX is dated November 3, 1971, Second Edition, June 12, 1972. Only in February 1973, with the advent of Third Edition, did `passwd` find its place among "user maintained commands." Fourth Edition (November 1973) was Third, rewritten in C; more importantly, it made its way out of the Labs in central New Jersey and the Bell System in Manhattan.

The first user outside of New Jersey was Neil Groundwater, now with Sun in Colorado. He was able to see utility in primitive systems, and in his words:

> The group I worked with at Whippany was "mechaniing" the analysis of some of the "outside plant" tasks involving ESS offices in New York City. The ESS machines generated call-failure messages . . . on teletype printers in the central offices. Although the UNIX host didn't communicate directly back to control the 'switch,' some artful wiring allowed a modem into the current-loop of the teletype printer and another modem at our office (on the 14th floor of 330 Madison Avenue in Manhattan) sent the signal into a multiplexer port.

By the way, Groundwater was also the first user of Ingres east of the Berkeley hills: he used it to track wiring changes in Manhattan.

Modems and telephone lines, of course, gave rise to further problems.

In October 1973, Dennis and Ken drove up to Yorktown Heights, NY, and gave a paper at SOSP. It was published in the July 1974 CACM. As I have related elsewhere (*A Quarter Century of UNIX*, 1994), the response was tremendous. Among other things, by mid-1974 both the University of Toronto and the University of Waterloo in Canada had PDP-11s. Tom Duff was an undergraduate at Waterloo and became a graduate student at Toronto. He told me how, in the spring of 1974, Doug McIlroy had visited to give a talk and how the students had later called Bell Labs "to grab the system source code."

McIlroy told me that he had dialed in from the Computer Club at Waterloo. "The UNIX phone number showed up on the bill" and so the students made use of it. Clearly a security breach.

The ARPANET 1970-1975

Bob Metcalfe, inventor of Ethernet and influential exponent of packet switching, summed up the problems of security on the ARPANET in RFC 602 – December 1973. Here it is, in its entirety:

> "The Stockings Were Hung by the Chimney with Care"
> The ARPA Computer Network is susceptible to security violations for at least the three following reasons:
> (1) Individual sites, used to physical limitations on machine access, have not yet taken sufficient precautions toward securing their systems against unauthorized remote use. For example, many people still use passwords which are easy to guess: their first names, their initials, their host name spelled backwards, a string of characters which are easy to type in sequence (e.g., ZXCVBNM).
> (2) The TIP allows access to the ARPANET to a much wider audience than is thought or intended. TIP phone numbers are posted, like those scribbled hastily on the walls of phone booths and men's rooms. The TIP required no user identification before giving service. Thus, many

people, including those who used to spend their time ripping off Ma Bell, get access to our stockings in a most anonymous way.
(3) There is lingering affection for the challenge of breaking someone's system. This affection lingers despite the fact that everyone knows that it's easy to break systems, even easier to crash them.
All of this would be quite humorous and cause for raucous eye winking and elbow nudging, if it weren't for the fact that in recent weeks at least two major serving hosts were crashed under suspicious circumstances by people who knew what they were risking; on yet a third system, the system wheel password was compromised – by two high school students in Los Angeles no less.

We suspect that the number of dangerous security violations is larger than any of us know is growing. You are advised not to sit "in hope that Saint Nicholas would soon be there."

I think that all of us may crack a smile here: crackable passwords, phone numbers on "Stickits" and walls, breakins by high school students? I'm shocked to find gambling going on.

In September 1971, there were 18 hosts on the ARPANET. At the time Metcalfe was writing, there were 31 host sites. In October 1974, there were 49. In January 1976, there were 63. A decade later (February 1986), there were 2,308; five years thereafter (January 1991), there were 376,000. This meant an enormous increase in users and telephone numbers. The exponential increase over the past few years has made the system yet more fragile in terms of security – or, better, insecurity.

More About UNIX

Maurice Wilkes had discussed password security as early as 1968 (Time-Sharing Computer Systems). The FIPS concerning DES appeared in the Federal Register (40FR12134) on March 17, 1975. Morris and Thompson published their "Password Security: A Case History" soon thereafter. It has appeared in many editions of the BSD documentation (It is © AT&T 1979). Dennis Ritchie's contribution was a brief article, "On the Security of UNIX" (also ©1979 and in the BSD documentation) and SUID – for which he holds US Patent #4135240, Protection of Data File Contents, January 16, 1979. Ritchie told me about the patent application:

> The patent was rejected at first because the examiner was unconvinced that the disclosure was complete and explicit enough for a person ordinarily skilled in the art to implement it. So I wrote a tiny, toy OS framework with UNIX protection modes less SUID, and we gave it together with the patent to a bright guy, new to UNIX, in the local Comp center. He successfully added SUID to the code, and a brief affidavit from him fixed the problem.

SUID (and SGID) still provide a way to grant users system access to which they are not otherwise entitled. This is extremely useful, but it also makes us insecure in other ways.

In 1984, Grampp and Morris wrote a brief article, "UNIX Operating System Security." It contains four "important handles to computer security":

1. Physical control of one's premises and computer facilities
2. Management commitment to security objectives
3. Education of employees as to what is expected of them
4. The existence of administrative procedures aimed at increased security.

Later they itemize a half-dozen points:

1. The insecure nature of passwords
2. Protection of files
3. Special privileges and responsibilities of administrators
4. Burglary tools and protection against them
5. Networking hazards
6. Data encryption

Also in 1984, Reeds and Weinberger published "File Security and the UNIX System Crypt Command." Here they pointed out, "No technique can be secure against wiretapping or its equivalent on the computer. Therefore no technique can be secure against the system administrator or other privileged users . . . the naive user has no chance."

In his Turing Award lecture in 1983, Ken Thompson described an insidious Trojan Horse using the C compiler as a tool that would miscompile the login command. I will refrain from further detail here.

A year ago, at the Internet Society meeting, Tsutomo Shimomura, the person who tracked down Kevin Mitnick, showed a video clip of the breakin by a hacker in Amsterdam to the Pacific Intelligence Center. The hacker went through the Stanford Physics Department. He got to the government computer and tried to ascertain who was on, which accounts have no password, and where the :: accounts were. There were six accounts; one was of the Submarine Squadron based at Camp Smith, Hawaii. The hacker then found a back door: an account called Dot.John. He ran it and got instant privileges – he'll never get another permission violation. It went on for a while, with Tsutomo shadowing him.

There was never a prosecution because the Netherlands didn't have computer crime laws in 1991. But the important thing is that he broke in successfully. Tsutomo mailed the videotape to the Pearl Harbor group (Pacific Command). When no response whatsoever was forthcoming for over a year, he, John Gage of Sun Microsystems, and Jeff Bear took it to Congressman Markey (1993). To the best of my knowledge, two more years have passed with nothing transpiring.

Tsutomo told us that the same Dutch hacker had broken into NASA (Kennedy Space Center) the previous day.

Our government (as well as the CIOs of many Fortune 500 companies) is in deep denial where security is concerned. Vint Cerf remarked to me that "A lot of those responsible for configuring hosts on the Internet . . . don't think hard enough about doing it right."

Conclusions

Multiuser systems have been around for more than 30 years. Problems with passwords have been with us nearly as long. The problems that arise as a result of our connecting computers with telephone wires have been with us for over 20 years. Despite the knowledge gained from the CTSS problem, despite the warnings of Metcalfe, despite the work of Ritchie, Thompson, Morris, Grampp, Reeds, and Weinberger, we arrived at the point where Sun was causing explosive growth in the use of UNIX and the Internet was beginning geometric growth, with only the merest improvements in security.

One of the holes exploited by the "worm" in 1988 was that of `rsh`. The very existence of an Internet within which one could remotely access a machine was one of the flaws.

Gene Spafford has pointed out that security comes at a cost. If we want our machinery and our data to be secure, we will have to surrender much of our flexibility and ease of access. We have had more than enough warnings, but despite the "worm," despite the German gang uncovered by Cliff Stoll, despite Kevin Mitnick, it is only within the past year or two that businesses and individuals have begun wholesale implementation of firewalls.

I have not gone through all this in order to be trite and tedious: I have done it because I am deeply concerned. I was consulting at a very large corporation in February where my host had her password taped to her workstation. "I can't remember those things," she remarked. A well-known programmer I lunch with uses his first name backwards with a + after it "because they insist on a nonalphanumeric character." Oh boy!

There is no point that was made by Metcalfe that I don't witness every ten days. Like folks who leave cameras on the seats of cars, we do not heed warnings and expose temptations to others. My New York apartment was burglarized in 1965. I have been very careful about my dwellings since then.

Good Software, Lousy Installation

by Scott Hazen Mueller
<scott@zorch.sf-bay.org>

One of the frustrations of dealing with computers for a living is installing third-party software. I can deal with the variations in quality; software from a multi-billion-dollar company darn well better install more easily than software from a one-person software house. It's those hidden assumptions that are the real killers – the slick GUI interface that won't run across a network or the space checker that falls over and dies if the columns in the `df` output run together. Herewith is my wish list for software installation.

Let My OS Go!

First and foremost, stay away from the operating system's files and partitions. At a minimum, don't fiddle with root `(/)` and `/usr`. Those belong to the OS vendor, not the application software. Any moderately sophisticated shop is probably going to upgrade its operating systems periodically, then guess what?

As with any rule, there are exceptions that owe to necessity. Networked applications frequently need to add port number definitions or service entries in the appropriate control files. But a robust application will be coded to operate even without those entries.

Start Me Up

Application start-ups should live in their own script files. This is another exception to rule one, and the damage is lessened by using private scripts. In other words, in a BSD environment, don't add your application startup to `rc.local` since you don't know what may be lurking there or whether it's even safe to touch.

In a System V-based setup, with `/etc/init.d` and friends, the typical case is just to add a new script anyway, so this is one case where I could say System V is "better" than BSD.

It Ain't Yours, So Don't Touch It

Don't assume you can use certain paths just because they're common to all systems of type "X." This is an extension of rule one. For example, many systems come with an empty `/usr/local` directory. This might seem like a pretty cool place to install, but developers can't know the policies of all the sites that might run their software. It might be reserved for public-domain software, it might not be in use at all, it might be mounted from a server controlled by another orga-

nization, or it might even be a world-writable playground for the site's own software developers.

Where the Buffalo Roam

Write your application so it can be installed anywhere. Asking users to set environment variables is OK; using more sophisticated techniques is even better. The less you assume about the path to the application, the easier it will be to install.

Take a Left and You'll Be Right

Corollary: don't assume that the install paths and execution paths are the same. Aside from magical file systems such as AFS (where the writable path is different from the read-only path), sites may mount their fileserver disks under `/export` (on the fileserver) and NFS-mount them at various other locations on the clients.

Local or Express

A well-written installer can cope with being run on one machine, reading media from another, installing the application onto a third, with output directed to a fourth, and the application will ultimately run on a fifth. I'll let you get away with the assumption that keyboard input comes from the same place the output goes to.

In even a small installation, the fileserver may be locked in a closet down the hall. Because system administrators value convenience, the CD readers and tape drives are rarely in the closet with the server, but they might not be on the desk of the administrator running that particular installation. In some cases, the administrator might even be running the installation remotely from another site, or your own company's support staff might be doing the installation as part of a service package.

A number of assumptions must be examined to do this right. If your application installs from CD, the install process should direct the administrator to mount and export the CD from the media server; the install process should be able to look for the CD files under an arbitrary path.

The install process might be occurring across a networked filesystem, so don't make assumptions about user "root" having write permission to the target directory. Many sites don't export "root" write permission on NFS-mounted disks.

Several system administrators believe that a system that does only one thing will do it very, very well. As a result, many sites dedicate systems as fileservers that will never run applications. Indeed, there are companies that have built entire businesses on selling boxes that do nothing but serve data via NFS. As a consequence, there is frequently a designated system that lives to run application-specific processes such as license servers. The odds are good that this box is distinct from both the media server and the server the installation was run on. The application installer needs to understand this case, or it needs to run on the license server while being remotely operated by the administrator. Ideally, the installer will handle both cases gracefully.

GUI or Gooey?

Always provide a non-GUI interface to your installer. People who know me know that I'm no fan of hand-holding interfaces. The reason is that fancy interfaces most frequently fail under the adverse conditions where you most need access to the systems hiding behind them. For instance, you might be offsite with limited access when you need to restart some application process. If the only access is via an X-based GUI, you would be completely out of luck. I'm not saying that the only interface has to be nongraphical, just that a nongraphical interface can be vital.

(Not) Like a Rock

Don't count on related packages being available during installation. The installer from a certain software company, which shall remain unnamed (but I sure wish I owned some founders' stock!), looks for their client application during the install process. Because I consider it silly to install software I don't need on single-function dedicated servers, the client software was naturally not installed. Fortunately, the installer failed gracefully, falling back by telling the user to start a client against some magic parameter.

Ham and Eggs

If your application needs another piece of software to operate, ship it on the media. The last thing I want to find out at the end of an all-day install process is that I'm supposed to download some other package off an FTP site and install that too.

However, if you do supply a copy of this other application, please indicate whether you've made any special tweaks to it. I might already have installed a bog-standard version of the same software, and now I'm wondering why your software doesn't work right.

The Fast Lane

Supply a set of defaults that ensures your application works. That way, novices can successfully install your application while still leaving all the flexibility in place for experienced installers.

A Hard Life

Lastly, deal with failures gracefully. I've had to go in and rewrite install scripts because they fail to deal with cases where the command output is bogus, but there's not really a problem. Check return codes – isn't that just good programming practice anyway? Sanity-check your input, and always, always give the operator the option to continue despite a seeming error condition.

Clip 'n' Save

1. First and foremost, stay away from operating systems files and partitions.

2. Application start-ups should live in their own script files.

3. Don't assume you can use certain paths just because they're common to all systems of type "X."

4. Write your application so it can be installed anywhere.

4a. Corollary: don't assume that the install paths and execution paths are the same.

5. A well-written installer can cope with being run on one machine, reading media from another, installing the application onto a third, with output directed to a fourth, and the application will ultimately run on a fifth.

6. Always provide a non-GUI interface to your installer.

7. Don't count on related packages being available during installation.

8. If your application needs another piece of software to operate, ship it on the media.

9. Supply a set of defaults that ensures your application works.

10. Deal with failures gracefully.

Security Policies – Lead-weighted Balloons or Safety Vests?

by Michele D. Crabb
<crabb@nas.nasa.gov>

One of the least favorite topics at many computer sites seems to be security policy design and implementation. However, well-defined security policies are a crucial element of any successful security infrastructure. They can also be the most difficult part of the framework to build, due to their controversial nature. In large organizations with many players, the security policy design and implementation process can often be a battlefield, pitting one group against another.

Yet the process need not be so difficult. A number of key points can aid the successful design and implementation of a security policy. These key elements, paired with an understanding of the organization's political environment, can make the difference between a security policy that serves as a safety vest and one that flies like a lead-weighted balloon.

Political Factors

The negative aspects of security policy design and implementation seem to stem mainly from political factors involved with policy design. Security policies affect everyone within an organization. People are human – and human nature varies greatly.

During my seven years enduring the challenges of a security analyst at a large supercomputing site, I have seen four trends that fuel most of the political turmoil.

The first is that many people strongly resist change. For example, say Peter and Mary are two happy, productive employees at New Age Software Solutions (NASS, an imaginary company). If Peter and Mary are in the habit of logging in nightly from their Internet service provider (ISP) account to their NASS accounts to perform some work functions when, suddenly, a policy is implemented to deny all network connections from ISPs, both Peter and Mary are going to be very unhappy. This change would require a change in the way they perform their work – perhaps even one that Mary or Peter would be unwilling to make.

This scenario brings me to the second trend, which is that many people tend to resist measures that impede (even in the most minimal way) their current level of productivity. I have witnessed many unhappy folks complaining about a simple change requiring them to log into a specific set of computers before logging into their personal workstation account, even though this procedure requires only an additional 30 to 60 seconds.

The third trend, which may be more common than the others, is that people generally don't like the "big brother" syndrome. Security policies often include logging and auditing events on computer systems. Such activities are sometimes viewed as an invasion of privacy from the user's perspective. Most people I have known and worked with have very strong opinions about privacy issues.

The last major trend I have witnessed is that some people, who have no strong opinion one way or the other, just like to "rock the boat." These folks tend to encourage controversy

among the ranks by playing the devil's advocate. In my seven years as a security analyst, I have seen many players take positions on all of these issues. At times, I have even been on different sides myself.

The basic elements that fuel the political fires are differing views and needs for security. People's views are often aligned according to their job responsibilities. Management is most often concerned about cost, company reputation, and integrity of data. Support staff are mostly concerned about getting the work done without a lot of unnecessary controls or roadblocks. Users are primarily concerned about sharing their ideas and sometimes their work, and they are also concerned about the privacy of their data. Getting all sides to agree on a single view or requirement is nearly impossible. Often you have to settle for an acceptable compromise.

There are also a few nonpolitical reasons that compound the difficulty of policy design and implementation. The first is that people often view security policies as large, monolithic documents that are difficult to follow and implement. In many cases, this observation may be true for poorly designed policies. I have found that technical people usually don't like to participate in bureaucratic tasks, such as policy design. Yet they are usually the key element in such a task. Lastly, I have found that people are frequently frustrated by the fact that what a policy should say and do is different from what it can actually accomplish – sort of the "fear of failure" syndrome.

These elements add up to a series of roadblocks that greatly impede the policy design process. However, there is hope for those faced with this challenging task.

Security Requirements

Many folks may ask, "Do we really need all of these security policies?" My answer would be a simple "yes." Without security policies, you don't have a security framework. I like to use the analogy of a society without traffic laws. For example, let's consider a four-way intersection. If there are no written laws or guidelines about what to do when approaching such an intersection, there will be a lot of accidents, arguments, and gridlocks. Some people will use common sense when approaching an intersection. They will slow down and make sure no other cars are coming before proceeding.

The same could be said for a computer site without written security policies. Many people will use common sense and try to do the prudent thing. However, there will be others who take advantage of the situation – sort of analogous to the intersection with no laws/guidelines. Some drivers will race through the intersection, assuming they have certain rights. This may work some of the time, but at other times, there will be accidents. Without written laws, how is it determined who is responsible for the accidents and other problems?

Security policies are important for other reasons, as well. The main purpose of security policies is to define the appropriate use of the computing resources and outline the penalties for misuse. If all users are working under the same assumptions and understanding, the computing resources will function in a more productive, secure, and useful manner. Hence, there will be a "level playing ground." Security policies often set the stage in terms of what tools and procedures are needed for an organization. For example, let's say our imaginary company, NASS, decides to implement a policy stating that remote network connections from all ISPs are to be denied. This would predicate the need to implement either a hardware or software packet-filtering solution.

Security policies are also needed because technical controls are not always available for the desired rules and regulations. Let's go back to our hypothetical example and say that NASS management has a strong desire to prohibit account sharing among the employees. When people share accounts, they often either provide their password to another person or they create an `.rhosts` entry for them. No technical controls can be implemented for stopping people from sharing their passwords with friends and family. Even if a one-time password system is in use, people can provide their token card/pin number to a friend or family member. In this case, the policy against account sharing, and management's support for taking actions against violators, is the only control available.

Key Design Factors

Policy design and implementation do not need to be a gut-wrenching battle pitting one side against the other. There are a number of key points that, when followed, can result in a "win-win" situation for all people involved. Policy design should be a team effort. You need to involve technical people who understand the ramifications of the proposed policy and management people who can – and want to – enforce the policy. You also need to involve user-level folks who understand how the majority of users do their work and how the policy will affect them.

After choosing your "team," you need to decide on the stance you want to use. There are two common approaches used for security policies:

1. "That which is not expressly permitted is prohibited."

2. "That which is not expressly prohibited is permitted."

Most folks prefer the second, less restrictive stance. Some environments, such as a highly classified computing site, might prefer to use the first stance. But for more open envi-

ronments, such as universities and many companies, the second stance is more appropriate.

The next key point is to designate a person or "body" to serve as the official interpreter of the policy. No doubt, at some point, there will be a controversy over the exact meaning of the policy. All such cases should be referred to the official policy interpreter, who has the final say.

Security policies need to follow some basic guidelines as well. They should be concise, easy to read, and written in everyday language that people are accustomed to hearing. People dislike large, monolithic documents full of technobabble that no one understands. People will neither read nor follow such policies. Security policies must also balance protection measures with productivity. This is a very important factor. The level of protection should be in alignment with the level of threat. Policies that either reduce productivity or make little sense (given the level of perceived threat) will not work. I have witnessed this fact firsthand.

Security policies must also be culturally acceptable within the organization and must be compatible with existing rules and regulations of the company. This is especially true for organizations that are part of larger entities. For example, let's say the different departments of NASS wish to implement their own policies regarding system auditing and logging. The Human Resources Department of NASS wishes to institute full-session logging for its employees to ensure they are not trying to access information for personal curiosity only. However, this is way out of line with the company stance that all employees shall be trusted and treated with respect and consideration while at work.

Finally, security policies must be updated on a regular basis to reflect the evolution of the organization. Policies are not a one-shot deal that, once published, never need to be revisited.

People are often tempted to put technical details in a security policy, but this is a poor idea. For one, technology changes too quickly. A policy containing technical details will most likely be out of date by the time it is approved and implemented. Additionally, many security controls cannot be implemented identically on different platforms. A policy that includes technical details will end up looking more like an implementation plan, which the general user community will have little interest in reading.

Finally, there are two very important factors regarding policy design. The first is that a security policy will often contain some risks that the organization is willing to accept. The second is that the resulting policy developed by the policy design team may not be exactly what the security analysts had in mind at the onset of the process. However, if the design team follows the key points discussed above, the policy design phase should proceed trouble-free for the most part.

Key Success Factors

Along with the policy design phase, there are some key factors to ensuring the success of the policy. These come into play during the implementation phase of the process. Prior to making a security policy an official document, provide a period for review and comments to all people affected by the policy. This helps gain the "buy-in" needed from the general user population to make the policy a success. It will also help catch any missed details in terms of unexpected impact on users.

As part of the implementation process, any necessary training required to ensure conformance to the policy should be provided to the staff and users. This should include the process of ensuring that everyone knows his/her individual responsibilities. For example, as part of the policy, you might want to make the user staff responsible for reporting suspicious activity, such as strange messages on their workstation consoles. If you implement such a policy, then you must train the users/staff as to what is considered "suspicious activity."

Hard copies of new policies should be distributed to all staff and users. Some organizations require people to sign a form stating that they have read the policy and will abide by it; I prefer this practice. You should also establish a policy-briefing program for new and temporary employees and make this part of your organization's new employee orientation program. Such measures will ensure that people get started on the right track. Ongoing policy awareness is also a good idea. You can provide policy "refreshers" whenever you feel they are needed. These can be in the form of posters, memos or email, policy awareness meetings, or an annual review and sign-off process.

Large organizations may also find it helpful to implement a simple method to request information regarding policies. This is usually in the format of a email alias, such as "security." You could even implement a "majordomo" type of list manager, which provides automated responses based on query type.

Ensuring the ongoing success of any security policy requires three crucial activities. First, people must be informed when the policy is effective. I have often found it very useful to inform management when our security policies have saved us from instances of certain doom.

Second, management must enforce the policies, and people should be informed when the policies are enforced. For example, let's revisit the account-sharing policy example implemented at NASS. If the employees knew that some peo-

ple were sharing accounts and they had never heard of anyone being reprimanded for violating the policy, the policy would basically be a failure and ignored by the general user population. Granted, most folks may never feel a need to share their account with their friends and family; however, those who want to can do so without the fear of punishment. But if users are informed whenever a policy is enforced, they are more likely to follow it.

The final element to ensuring the ongoing success of any security policy is to implement an annual review phase where all members of the user/support community have an opportunity to review and comment on the policy and proposed changes. Security policies should evolve with the organization. What is appropriate or feasible one year might not be so the next year.

Security policy design and implementation are not easy processes. They are politically charged and challenged by many roadblocks. However, I have tried to point out some areas that are key to smoothing out the design and implementation phases.

If you are interested in additional information regarding security policies or are looking for some good examples, try these Web sites:

<http://www.gatech.edu/itis/policy/usage/contents.html>
<http://csrc.ncsl.nist.gov/secplcy/>
<http://www.eff.org:11/CAF/policies>
<ftp://nic.ddn.mil/rfc/rfc.1244.txt>

More Secure Mnemonic Passwords: User-Friendly Passwords for Real Humans

by Stephan Vladimir Bugaj
<stephan@cmdesigns.com>

No matter how many expensive security measures you have on your network, if you have users with passwords, then these passwords are the most vulnerable part of your network. There are various complex or expensive solutions (such as one-time passwords) that are applicable in some situations, but the fact remains that many people will use vanilla password systems on their machines and that they will use "dog" as their password if given the opportunity.

It's simple to enforce minimum password lengths, character requirements, and password aging; but these are not quite enough. You either wind up with passwords like "2dogdog" or, if you follow most expert recommendations, you get something like "v3%~/B1+bQ}", which nobody will ever remember. What is going to happen with "v3%~/B1+bQ}" is obvious: users are going to write their password down on a Post-it and stick it to their monitor.

There are a lot of discussions about how to strike a balance between the security of random passwords and the security of passwords that are simple enough to be memorized. Simple passwords are too easily cracked, so random passwords are generally preferred (though not necessarily generally used). I suggest that administrators encourage what some people (like myself) already do when creating passwords: employ mnemonic devices to make the pseudorandom passwords that I call More Secure Mnemonic Passwords.

MSMPs are passwords which are derived from simple passwords that the person will easily remember but that use mnemonic substitutions to give the password a more random quality. These passwords are easily constructed and easier to remember than random passwords. Because they follow an obvious pattern, they are more easily cracked, but imperfections abound in any security system. Here are a few examples of construction MSMPs:

```
batman → b@man →b@mAn → b@m4n
cheese → ch33s3 →ch33s3!!
chillywilly → chillywilly → ch!llyw!lly
```

In constructing MSMPs, you can use both phonetic and typographic substitutions. The "@" in `b@man` sounds like "at," whereas the "4" in `b@m4n` looks like a capital "A." It is possible to make a formalized set of these substitutions for use in an organization, but this will make it more difficult for individual users to create meaningful (to them) MSMPs.

MSMPs can also be constructed using mental associations and logical (or illogical) conclusions drawn from the original password idea. The original concept can be a word or a phrase. Formal rules can be made for substitutions, but it increases the security and memorability if you just encourage users to follow their mnemonic instincts. Here are a few more complex MSMPs that undergo both mnemonic character substitution and associative morphing:

```
Dr. Seuss →the cat in the hat → thec@intheh@
        → 1c@=>1h@
i live for UNIX → !l!ve4unl* →
        @^#&%~c0reDump!
i voted for Pat Robertson →
        iv0ted4P@R0berts0n → !me=666!
fnord → fn0rd → ?23skid00?
```

Passwords are susceptible to several kinds of attacks. The level of knowledge about your users is a major factor in many kinds of attacks. User knowledge is a big advantage for an attacker, and knowing the name of someone's pet is almost as good as knowing where they keep their Post-its

with their passwords on them.
With MSMPs there is a lot more work involved in being able to guess passwords based on background knowledge of the user's life. It may be a trivial effort to find out that someone has a dog named Duke, but it would take a lot more effort to follow this line of reasoning:

i have a dog named duke
i got the dog from the ASPCA while living in New York
my uncle Al lives in New York
uncle Al liked the Mets, especially Darryl Strawberry
therefore my password is `$tr4wb3rry`

Whether or not this is a ridiculous line of reasoning, the general idea is to encourage users to make deeper associations in creating their passwords and then to make mnemonic character replacements when spelling out the resulting word.

This scheme should give your users' passwords better protection against both manual and automated attempts to break them. They are not as secure against automated attacks as truly random passwords; but in a system where truly random passwords are viable, user sophistication will be at a level where even more complex precautions may be viable. However, if you have unsophisticated or numerous users, this scheme should help mitigate what I like to call "the ridiculous password problem."

This is no single solution for all your password problems. Indeed, in our organization, we have people who can't remember any passwords at all, and even those who can are still susceptible to social engineering. Nothing is a substitute for a comprehensive security plan and good understanding between administrators and users, but I hope this system can help you develop both. Besides, it's a fun way to make new passwords.

Tcl, CGI, and Security

by Qusay H. Mahmoud
<k3is@unb.ca>

This article shows how to write CGI scripts with the scripting language Tcl (Tool Command Language). The article assumes some familiarity with CGI and Tcl.

A CGI script is invoked by an HTTP server, usually to process user input submitted through an HTML `<FORM>` tag.

The output of a CGI script should consist of two sections, separated by a blank line. The first section contains a number of headers telling the client what kind of data is following. Tcl code to generate a minimal header section looks like this:

```
puts "Content-type: text/html\n"
```

The second section is usually HTML, which allows the client software to display nicely formatted text with headers. Here is a simple Tcl code that prints a simple piece of HTML:

```
puts "<HTML>"
puts "<TITLE>CGI Output</TITLE>"
puts "<BODY>"
puts "<H1>This is my first Tcl example</H1>"
puts "Hello, world!"
puts "</BODY>"
puts "</HTML>"
```

This is simple. However, in writing CGI scripts to process forms, there is more to worry about. Basically, two methods can be used to access your forms. These methods are `GET` and `POST`. The method you use will determine which form of the encoded results you will receive. It is recommended to use the `POST` method, however.

If your form has `METHOD="GET"` in its `FORM` tag, your CGI script will receive the encoded form input in the environment variable `QUERY_STRING`. If your form has `METHOD="POST"` in its `FORM` tag, your CGI script will receive the encoded form input on `stdin`. The server will not send you an `EOF` on the end of the data: instead, you should use the environment variable `CONTENT_LENGTH` to determine how much data you should read from `stdin`.

When you write a form, each of your input items has a `NAME` tag. When the user places data in these items in the form, that information is encoded into the form data, which is a stream of `name=value` pairs separated by the `&` character.

The basic procedure is to split the data by the `&` character. Then, for each `name=value` pair you get for this, you should URL decode the name, and then the value, and then do what you like with them.

Fortunately, there are already a number of programs that will do this decoding for you. The following is a simple Tcl script that will do the decoding; actually, it also prints all the environment variables. The script was originally written by *<robert.bagwill@nist.gov>* and modified by *<dl@hplyot.obspm.fr>*.

```
#!/usr/local/bin/tclsh
# tcl-cgi2.tcl
# Decode CGI/1.1 parameters, args, and
# environment variables
#
# Rewrite by dl@hplyot.obspm.fr - v2.2
# based on robert.bagwill@nist.gov's
# version
#
# no warranty, no rights reserved
#
set envvars {SERVER_SOFTWARE SERVER_NAME
GATEWAY_INTERFACE SERVER_PROTOCOL
SERVER_PORT REQUEST_METHOD PATH_INFO
```

```
PATH_TRANSLATED SCRIPT_NAME QUERY_STRING
REMOTE_ HOST REMOTE_ADDR REMOTE_USER
AUTH_TYPE CONTENT_TYPE CONTENT_LENGTH
HTTP_ACCEPT HTTP_REFERER HTTP_USER_AGENT}

# UnCgi Translation hack, in Tcl, v1.4 5/1995
# by dl@hplyot.obspm.fr

proc uncgi {buf} {
regsub -all {\\'} $buf {'} buf ;
regsub -all {\\} $buf {\\\\} buf ;
regsub -all { } $buf {\ } buf ;
regsub -all {\+} $buf {\ } buf ;
regsub -all {\$} $buf {\$} buf ;
regsub -all \n $buf {\n} buf ;
regsub -all {;} $buf {\;} buf ;
regsub -all {\[} $buf {\[} buf ;
regsub -all \" $buf \\\" buf ;
regsub ^\{ $buf \\\{ buf ;
regsub -all -nocase {%([a-fA-F0-9]
    [a-fA-F0-9])} $buf {[format %c 0x\1]}buf
eval return \"$buf\"
}

# returns in the 'cgi' array all the
# parameters sent to the script through
# 'message' (each array cell is a list (ie if
# only one value is expected through 'test'
# variable, use [lindex $cgi(test) 0] to get
# it)).

proc parse_cgi_message {message} {
global cgi;
set cgi() "";
foreach pair [split $message &] {
    set plst [split $pair =];
    set name [uncgi [lindex $plst 0]];
    set val [uncgi [lindex $plst 1]];
    lappend cgi($name) $val;
}
}
puts "Content-type: text/html\n"
puts "<HTML>"
puts "<HEAD>"
puts "<TITLE>CGI/1.1 TCL script report:
       </TITLE>"
puts "</HEAD>"
puts "<BODY>"
puts "<H1>Command Line Arguments</H1>"
puts "argc is $argc. argv is $argv."
puts ""
puts "<H1>Message</H1>"

if {[string compare $env(REQUEST_METHOD)
       "POST"]==0} {
    set message [read stdin
       $env(CONTENT_LENGTH)];
}
else {
    set message $env(QUERY_STRING)
}
parse_cgi_message $message;

puts "<DL>"
foreach var [array names cgi] {
    puts "<DT>$var<DD>$cgi($var)";
}
puts "</DL>"
puts "<H1>Environment Variables</H1>"
puts "<DL>"
foreach var $envvars {
    if {[info exists env($var)]} {
        puts "<DT>$var<DD>$env($var)"
    }
}
puts "</DL>"
puts "</BODY>"
puts "</HTML>"
```

There are other programs available on the Internet that you can use to decode your data. For more info on these programs, please take a look at:

<http://www.hyperion.com/~koreth/uncgi.html>
<ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/cgi/tcl-proc-args.tar.Z>

Tcl and all other scripting languages are unsafe for writing CGI scripts, so special care should be taken when writing CGI scripts. For example, if you invoke an external program via the exec system call, make sure you don't pass arbitrary strings received from the client to the shell. This is a well-known security hole whereby intruders can exploit the CGI script to invoke arbitrary shell commands. If you must pass a string you have received from a form to a shell command, then you should make sure the string contains only alphanumeric characters, dashes, underscores, and periods.

Try to avoid using the `eval`. `exec`, `uplevel` and such commands in your CGI scripts. For example, if the following is contained in a CGI script:

```
proc hack {args} {
    # script body
}

eval hack $argv
```

an intruder can execute the CGI script with the argument:

```
http://domain.com/cgi-bin/hack?exec
       rm -rf /
```

and the command `exec rm -r /` will be executed by `tclsh`.

Fortunately, Tcl provides a way to create safe interpreters where you can not invoke `exec` or open files or do other stuff that is considered unsafe. The following program which was

posted on comp.lang.tcl by *jyl@noam.eng.sun.com* illustrates how you can use Tcl safely for CGI scripts:

```
#include        <tcl.h>

int
main(argc, argv)
int argc; char **argv;
{
    Tcl_Interp *interp;

    if (argc != 2) {
        exit(1);
    }

    interp = Tcl_CreateInterp();
            /* Creates an interpreter. */
    (void) Tcl_MakeSafe(interp);
            /* Makes it safe for scripts. */

    /*
    * Assume that the CGI script is in a file
    *and that we are being passed the file
    * name. We simply eval the file and exit
    * with 0 on success, and 1 on failure.
    */

    if (Tcl_EvalFile(interp, argv[1]) ==
            TCL_OK) {
        exit(1);
    }
    exit(0);
}
```

Compile this program, and place it where the CGI mechanism expects to find executables. Name it `tclsh` or whatever the script expects. You might want to make more commands available in the safe interpreter or make CGI-specific commands available to enable the script to do interesting things.

Another way to write safe CGI scripts in Tcl is to use the new scripting language called NeoScript. It is based on Safe-Tcl. NeoScript is a programming language that allows both simple and complex programs to be embedded into HTML files.

NeoScripts execute in a safe Tcl interpreter, a restricted environment that prevents NeoScript programs from doing malicious things like running arbitrary commands and deleting files. They thus provide an excellent way for Web masters to allow their users to write CGI and server-side include programs on their Web servers without incurring the risk that the users will, either accidentally or on purpose, allow the system to be subverted by outsiders.

In addition to the command set provided by Safe Tcl, several new commands and variables have been added to provide the HTML/NeoScript author with a number of simple, application-oriented web page services.

For more info about NeoScript, please see: *<http://www.neosoft.com/neoscript>*.

An Update on Standards Relevant to USENIX Members

by Nicholas M. Stoughton
USENIX Standards Report Editor
<nick@usenix.org>

The Decline and Fall of POSIX?

by Nicholas M. Stoughton

The March Internet Engineering Task Force (IETF) meeting in Los Angeles attracted well over 1100 attendees. They ranged from those who wanted to find out more about the Internet (i.e., were using it as an educational vehicle), to interested observers wanting to report back latest developments to their companies, through a number of real developers, actively building new standards.

Meanwhile, the IEEE Portable Applications Standards Committee (PASC) continues to have a declining membership, now down to well under 100 for the last three meetings (actually 64 at the most recent meeting).

The IETF churns out large numbers of standards, with each working group producing two or three a year. PASC has slowed to a trickle, or so it seems. Actually, looking at the output of the two groups side by side, the picture is not nearly as dark as it first appears.

The average IETF standard is 30 or 40 pages long. It is written by engineers for engineers. Few application developers need worry overtly about the nitty gritty of the lower levels of the protocols between the parts of their application, so long as they work. Most IETF standards have working reference implementations to assist in describing behavior. Therefore, the language in the standard need not be as precise as required by a POSIX standard.

The average PASC standard is 300 or 400 pages of extremely carefully reviewed material. It takes between three and six years to develop one of these beasts. Its audience is a much wider proportion of application developers and system implementors. It must be couched in unambiguous terms, and not rely on anyone's "common sense" to determine the meaning of what is written.

IETF standards are built on "rough consensus" and "working code." Rough consensus is not actually defined anywhere, and it is often up to the working group chair to see how many heads in the room are nodding; "a good hum" is all that's needed. It then goes through various subsequent levels of inspection and implementation before becoming a full standard. The working code is often the arbiter in cases of interpretation; if your implementation can talk to mine, then it's probably OK.

PASC standards on the other hand have a very tightly defined idea of consensus. The rules are rigidly applied. The ballot groups are often large (POSIX.1a for example has 127 members) who have a small finite window to vote on a particular draft. If 75% of the respondents approve (and 75% of the group has to respond), then the draft is approved. Most large standards go through about 12 drafts. IETF documents (Requests for Comment, or RFCs) probably go through four or five before they become "draft" standards.

The following reports are published in this column:

- POSIX
- Web Portable Applications Study Group
- NIST CSL
- POSIX.1a: System Interface Extensions
- POSIX.1k: Removable Media
- Internet Architecture Board (IAB)

Our Standards Report Editor, **Nick Stoughton**, welcomes dialogue between this column and you, the readers. Please send any comments you might have to *<nick@usenix.org>*.

innovation be left for future standardization (once it reaches the status of existing practice). The holders of the Intellectual Property Rights (IPRs) must be engaged, and supportive, this is not to "standardize Java in spite of Sun," it is to acknowledge that Sun (and Netscape, Microsoft, et al.) have developed very useful contributions in the area of portable applications, and that standards will establish increased consumer confidence, improved industry acceptance and commitment, and a foundation for profiles for Information Infrastructure environments.

We are looking for first version of standards in this area to enter balloting in 1997. By keeping the focus on the core of what exists, and engaging the IPR holders, we should be able to generate drafts quickly, and produce fairly "small" standards. One advantage of getting in "early" on these specifications is that they are not as complex as they will be in the future. We may want to consider a "drop dead" date for completion of the work (REVCOM submission), perhaps 2 years from the date of Project approval.

The study group would meet July 17 and 18 in Nashua, New Hampshire, and either generate draft Project Authorization Requests (PARs) and schedule further meetings or determine that this work should not be pursued at this time. It is possible that one or more projects here might be targeted for use of an accelerated process, with additional meetings involved. Also, one or more of these might seek to utilize an "organizational representation" model (emerging out of current discussions at the IEEE level) as opposed to the individual expert model traditionally used in PASC.

Jim Isaak will convene the Study Group, and invite participation though direct contact with interested parties, ANSI IISP and related channels of communication, and publicity releases though IEEE and the Computer Society.

Attendees will be expected to pay a $200 meeting arrangement fee including lunch for the two days, or they could preregister for the full week with lunches at $300.

The Death of the NIST Computer Systems Lab

by Stephen R. Walli
<srw@softway.com>

Events have occurred recently in the NIST Computer Systems Lab that are very disturbing, and could have a number of implications for those of us that purchase systems as part of our jobs. NIST, the National Institute for Standards and Technology, is part of the Department of Commerce. (It was formerly known as the National Bureau of Standards.) The Computer Systems Lab (CSL) is the group that is responsible for producing Federal Information Processing Standards (FIPS) to aid in government procurement, certification programs to support the FIPS that are widely recognized in the private sector as well as the US federal government, and represents one of the primary centers for computer and information processing standards and specifications expertise in the US federal government. Instead of sticking to this core competency, however, there is a new world order that not only ignores but destroys the good work done to date, and further ignoring history, boldly wanders off into the weeds.

A FIPS is a simple document that acts as a government procurement reference to another standard. Its usefulness is probably best understood by example. The IEEE POSIX.1 standard defines the programmatic interface to operating system services, based on the historical UNIX system. In its conformance section, the standard claims that implementations must provide all the services described in the standard, that they must behave as described, and describes a set of documentation requirements that become a system's POSIX.1 Conformance Document. The standard has a number of implementation options, limits, and there are a number of places that are labelled implementation defined or unspecified. Therefore two different systems could both implement different options (for example one supports job control, and the other doesn't), but still claim conformance to the standard.

NIST FIPS Pub 151-2 is the POSIX.1 FIPS. It incorporates the text of the IEEE POSIX.1 standard by reference, then further identifies what optional functionality is required, what limits need to be set, and so on. Thus there is better consistency with respect to the POSIX.1 standard across systems that meet the FIPS 151-2 criteria, and therefore the source-code portability model across the machines is more consistent.

There are FIPS for ANSI C (FIPS 160), the POSIX.2 Shell and Utilities (FIPS 189), SQL, Fortran, and so on.

To ensure systems claiming conformance to a FIPS actually meet some minimum requirements, NIST CSL puts testing and certification processes in place wherever possible. Using the POSIX.1 FIPS as an example again, a system vendor takes their implementation to an accredited testing lab, who runs the NIST POSIX.1 Conformance Test Suite (PCTS), and those results along with the conformance document are submitted to NIST. The results are validated, and if all is well, a certificate is issued acknowledging that the system has run the test suite, and the test results have been inspected.

The POSIX.1 test suite has its roots in the original System V Verification Suite and has been evolving since 1988 when the original suite was built. It is a large and robust test suite. The process and policies have also been continuously evolving and improving over time.

While the FIPS 151-2 document is essential to government procurements requiring POSIX-like services, the FIPS 151-2 certificate has broader recognition and visibility in the commercial world as well as a reasonable "Good Housekeeping

So, in fact, the POSIX process is roughly equivalent in output to the IETF, but the resulting output is far more formal a document. The reaction time – the time from idea to something concrete – is a major problem, but the rate at which work gets done is not substantially different.

But the shrinking size of PASC has other, more sinister, connotations. Does a working group of four or five people really represent the industry as a whole? Isn't some commercial concern, such as X/Open, doing a better job, faster, with more representation, than PASC?

I would argue strongly against this idea. I hope you will agree with me, too!

The strength of PASC lies in the fact of its openness, and its cross-industry participation. X/Open, while having a User Council, is basically funded and run by (and for) the vendors. It does a fantastic job for them, and I am not in any sense knocking that. PASC, on the other hand, is not about the Working Groups themselves, but the wider audience of people involved in building standards – the ballot groups in particular. Anyone can join a ballot group (even non-members of the IEEE, though their ballot need not be regarded as binding). Anyone can influence a POSIX standard provided the consensus agrees. End-users of the standards, both application developers and the implementors, are able to voice their opinions, and can try to ensure that the result is useful and acceptable. I can't do that to the Single UNIX Specification. But I can (and have) influenced the content of POSIX.1, for example.

The debate on the declining attendance of PASC meetings is raging around me as I write this. The three options we are discussing are

- Make no changes, we are doing OK
- Stop developing new POSIX standards, our job is done there. Change PASC to become purely a care and maintenance group for the finished work.
- Alter PASC's scope to take in other, sexier, work and attract more people

That third option may have some merit. But PASC has done a good job in defining application program interfaces for portability. To branch out into other areas, like languages or protocols, is not only a step into the unknown, but also treads on the toes of other existing groups.

The current scope of PASC is based on the interfaces needed to develop portable applications. They describe the interface between the application and the environment in which it is to run. The real question is how much has that environment changed in the last 12 years or so that has been working. Are we only working a small percentage of our scope?

As a part of this debate, a new study group is to form at the next meeting to look at some new functionality that might be within our scope. A notice for this group follows below.

The debate will continue for some months, I'm sure. Please feel free to mail me with your opinions.

Web Portable Applications Study Group

Jim Isaak <isaak@ljo.dec.com> reports on the proposed Web Portable Applications Study Group

At the April meeting of the IEEE Portable Applications Standards Committee (PASC), a new study group was formed to investigate a new environment for applications portability that is emerging: World Wide Web based applications portability. This is an area that is directly within the scope of PASC, is developing significant implementation experience, and reflects a key area for future application portability standards. This is also an area of work identified by the ANSI Information Infrastructure Standards Panel (IISP) activities. One requirement specifically calls for interfaces for application portability in an architecturally neutral form. A number of technologies for Web based applications meet this objective.

Web based applications are an emerging technology, and there are sure to be "bugs" that will surface over the next year, together with extensions and new facilities that will emerge. What is possible right now is to define and document the stable core of the existing practice in this area, and set the stage for extension and innovation.

The purpose of the study group is to determine what standardization is appropriate at this time, and how that work should progress (hopefully in IEEE/PASC).

Existing practice in this area that should be drawn upon to direct standards include: Java, JavaScript, and VBScript (Visual Basic Script). Key players in this include Sun Microsystems, Netscape, and Microsoft. It is possible that we might see projects in all three of these areas based on the existing practice in each area. We may also see multiple projects appropriate for work on something like Java, with the language specification, class library specification and byte-code specifications representing three different, but important interfaces.

Ground Rules: This must *not* become a repeat of the now infamous "GUI" wars (where there were two directly competing groups fighting to make their product the standard). The objective will be to standardize existing practice, not to attempt to integrate or force alternatives into a single solution. Multiple alternative standards must be permitted and even encouraged. The scope of the work should be clearly focused so that changes from existing practice reflect correcting bugs (such as security gaps), that extensions and

Seal of Approval" that the system of interest meets certain fundamental criteria. It is a reasonable hurdle that supports a level playing field approach to systems procurement regardless of whether the purchaser is a government or commercial organization.

There has been a long history of success with the testing programs. (A USENIX conference speaker once made reference to the original NIST FORTRAN compiler test suite with awe and respect, figuring that it had been written by a group of pedantic grandmothers sucking lemons.) As there has been a need for further test suites faster to support the rapid progress of technology, and with the recognition that good test suites cost money, NIST has adapted to the changing times. As it has become too expensive and slow to develop the test technology in-house, NIST personnel worked with industry to adapt and adopt appropriate test suites.

To build a reasonable and representative FIPS and its attendant certification program requires experience and understanding. Government organizations that mandate things without understanding them tend to create white elephants that finally collapse under their own weight. Sticking to the POSIX FIPS example, the NIST CSL has actively participated in the community, developing the standard, such that FIPS 151-2 and the PCTS were well balanced and accomplished their goals. Indeed, NIST CSL's participation dates back to the /usr/group 1984 standard.

There has been exactly once that NIST has thrown its weight around with respect to POSIX, and that was the introduction of the original FIPS 151 in 1988 just prior to the final close of the IEEE POSIX.1 standard itself. Again, this decision was likely based on experience with the standards development body, and NIST's understanding of the stableness of the standard itself. (Both IEEEix and the IEEE Trial Use POSIX.1 standard were "cooked"). NIST CSL personnel have apparently always been sensitive to the fact that they are part of the Department of Commerce, and that while they may want to mandate certain things for US federal procurements, they also need to support an economy full of companies and employees that work and pay taxes.

While this work lacked sex appeal, it was essential and responsible. And one would expect the government to do the essential and responsible, supporting the economic needs of the government procurement agencies, while indirectly contributing to the industry as a whole, rather than looking for sex appeal.

In the blood-sport of re-organization at the Department of Commerce, mandates have been changed and personnel reorganized to new missions within the NIST CSL. Somewhere, someone has decided that all the essential work that has gone on to date is no longer necessary or relevant.

The personnel in the core of the FIPS program and responsible for certifications have been re-assigned to new programs. There is apparently a single person left with the authority to sign FIPS certificates, but that person has no idea how few weeks or months they may be allowed to continue in this role, as they are essentially doing this of their own volition. The new direction (or lack thereof) feels that this work is no longer needed and that *the industry will police itself!* Such naivete is stunning!

It has even been suggested that Manufacturer Declarations would suffice. Systems vendors will be able to stand up and say: I conform to that FIPS – trust me. There have been a number of occasions (some recent) where I have spoken to vendors that claim to be POSIX-conforming, but have been unable to show even a conformance document, let alone a FIPS certificate. And being saddled with a situation where the user claims something is broken in an implementation with respect to the standard is a terrible case of closing the gate after the horses have escaped. At best, responsive vendors guided by the size of the customer may fix it for the next release.

It will be interesting to see how long it takes for history to repeat itself, as it's being ignored, and the demand for these certification programs to be established occurs again. It will also be interesting to see how long it takes for CSL personnel to come up to speed on the specifications, standards and programs the next time around, as presumably the current expertise will have been lost through re-assignment and attrition by the time someone realizes things are broken and need fixing.

One might be able to understand this ugly amputation if it was naively done for cost savings and budgetary reasons. The worst part of this is that this is more of a mandate change, walking away from a role with a proven successful track record, towards a role with a proven failed track record. People just have not read their history books as they created this bold new mandate.

Essentially, the resources of the NIST CSL are being turned towards developing and delivering new fundamental technologies to industry. (No, really, that's what they want to do.) One acronym example of the government success rate here: PCTE (the Portable Common Tool Environment). Here the government attempted to "encourage" a technology that they believed was fundamental, and that the industry was just not supporting. They built and delivered a reference implementation. There was even work on an ISO standard for this, (but then there's also an ISO standard for BASIC.) PCTE still hasn't taken the industry by storm, and still isn't accepted as a particularly viable technology in the real world. What tools exist are expensive. By fiddling around with the technology, supporting something they believe is goodness and light but has no industry support, the government created a false econ-

omy as there is always someone willing to build expensive tools for government demand. In the new world order of the CSL, there may not even be a government market for new "fundamental" technologies, and it will be a complete waste of effort.

Forcing a technological model without the economic support of a real market place has failed on other occasions. There was supposedly a point where the installed base of TCP/IP was growing faster than the demand for OSI. Economic reality finally settled in after who knows *how* much money was sunk into the OSI white elephant.

So hat's off to the team of people that have done so much useful work in our community. They and their efforts will be missed. Let us hope not too much damage is done before their efforts are appreciated at least in hindsight by their employer.

R.I.P. NIST Computer Systems Lab.

Report on POSIX.1a: System Interface Extensions

J. Jay Meyer <jjm1@rsvl.unisys.com> reports on the April 14-19, 1996 meeting in Jackson Hole, WY

The System Services Core Interfaces working group worked on draft 12.5 of POSIX.1a that was produced very recently. During the meeting, all of the remaining ballot objections were dealt with. Either they were already addressed, or we knew what to do. We also looked through the interpretations for things that could be clarified and haven't been so far.

At the last meeting we identified that the rationale for the trailing slash change needed to be rewritten and turned into permanent rationale. This, and many other issues were taken care of between meetings through collaboration of the Technical Editor and Ballot Coordinator (way to go, Lee and Nick!).

Issues related to flush of input files and atomicity of file operations were more clearly defined during the meeting, and we decided what to address and what not to address, and updates related to this issue are completed as well.

In general, we are very close to having a ballot-ready draft, the restricting factor being, as always, the available time of the working group members after we return to our "day-jobs."

The POSIX.1a document is moving all of the C language functions formerly described in POSIX.2 into POSIX.1a. Near the end of the meeting we still seemed to have problems in the area of divorcing the POSIX.2 function descriptions in POSIX.1a from the POSIX.2 standard. This issue may be the biggest problem in our next ballot round and/or a barrier to getting the new document produced.

In the POSIX.1a working group and in the System Services plenary, we have begun to discuss the scope and purpose of the POSIX.1 revision project that we will soon need to begin working on. Some of the material that this project could work on include adding more named options, converting to one header per function (where it can be done and makes sense), interfaces to solve inadequacies in the way that error information is returned, and questions that came from interpretations that have thus far eluded good answers. We intend to collect the important remaining mail exploders from the various system services groups and place them on the IDI machine. We hope to make the System Services policies available on the public areas of the IEEE Standards Process Automation (SPA) system (*http://stdsbbs.ieee.org*), and to experiment a little with using the private areas of the SPA system to help us do our work.

Report on POSIX.1k: Removable Media

Chuck DeBaun <debaun@fnal.fnal.gov> reports on the April 14-19, 1996 meeting in Jackson Hole, WY

The Removable Media group was formed to create an optional standard for a removable media resource management command line interface. However, in the search through existing standards, it was noted that such a standard could not be implemented in a strictly compliant POSIX environment. Indeed, serial media, better known as tape, cannot be supported in such an environment.

Thus, as a first step, the Removable group submitted a Project Authorization Request (PAR) to provide the missing `mtio` semantics in POSIX.1, so that serial media, a primary type of removable media, can be supported in a POSIX environment. This PAR was approved in June 1995. A proposal is currently on the table and is being used as a working document. Draft 5 of this proposal is expected following the April 1996 meeting.

At the same time, a PAR (POSIX.2e) was accepted to provide the `mt` utility definition for POSIX.2. This will provide a command line interface to the `mtio` function. A draft proposal was presented and accepted as a working paper at the April 1996 meeting.

A third PAR is being prepared for the actual removable media resource management command line interface specification. This work is being delayed by the need to create support for serial media before it can be started. The need to backtrack to create the `mtio` semantics has caused a marked decrease in interest and attendance further delaying action in this area. This group is currently seeking an implementation upon which to base the work.

Despite the low attendance, significant work was done at the April 1996 meeting tightening up the verbiage in the

POSIX.1k proposal. The POSIX.2e draft proposal was accepted as the working paper for the `mt` work.

What Does The IAB Do Anyway?

Brian Carpenter <brian@dxcoms.cern.ch>, IAB Chair, presents an introduction to the work of the Internet Architecture Board. This report is reprinted with permission from ConneXions. *(ISSN # 0894-5926)* ConneXions – The Interoperability Report *is published monthly by: Interop Company, a division of SOFTBANK Expos.*
Subscription hotline: 1-800-575-5717 or +1 610-892-1959
URL: http://www.interop.com
Free sample issue and list of back issues available upon request.

The "Internet Architecture Board" (IAB) sounds as if it is something rather grand, perhaps consisting of a group of people in formal business clothes, sitting around an impressive oak table, under the watchful eyes of an oil painting of The Founder of the Internet. The reality is rather different, and this article is intended to give a feeling for what the IAB really is, what it does, and equally important what it cannot do.

First, The Formalities

The IAB was originally called the Internet Activities Board, and it was set up in 1983, chaired by Dave Clark, back in the days when the Internet was still largely a research activity of the US Government. The early history of the IAB is hard to trace in detail from the public record, for a reason expressed clearly in the minutes of its meeting in January 1990: "The IAB decided that IAB meeting minutes will be published to the Internet community." The earlier minutes are not on the public record. A good snapshot of the IAB in 1990, and a short history, are given in RFC 1160, written by Vint Cerf who was the second IAB Chair. He was followed in this post by Lyman Chapin and Christian Huitema. In any case, the 1980s are prehistory as far as the Internet is concerned, and this article concentrates on the present.

Today, the IAB consists of 13 voting members. Of these, six are nominated each year by a nominating committee drawn from the Internet Engineering Task Force (IETF), for a two year term. This membership has to be approved by the Board of Trustees of the Internet Society. Indeed, one of the main motivations for the foundation of the Internet Society was to provide a legal umbrella for the IAB and for the IETF's standardization actions. The thirteenth voting member of the IAB is the IETF Chair.

In addition, IAB meetings are attended by a representative of the Internet Assigned Numbers Authority (IANA) and of the RFC Editor, by a liaison with the Internet Engineering Steering Group (IESG), and by the Chair of the Internet Research Steering Group (IRSG). Finally, the IAB has a volunteer Executive Director. The IAB elects its own Chair from among its 12 IETF-nominated members.

Now, What Are The Meetings Really Like?

Most of the meetings take the form of two-hour telephone conferences about once a month. Due to time-zones, it is early morning for members on the US West Coast, late afternoon for Europeans, and after midnight for our Australian member. Those on the US East Coast have a comfortable mid-morning time slot.

Of course, on a telephone conference, it is hard to see whether the others are wearing smart business suits. In our face-to-face meetings, it's pretty obvious that most of them are not. These meetings usually take place at IETF locations, three times a year. Unfortunately, although we are all in the same time-zone physically, it is guaranteed that some of us will be jet-lagged at every meeting. So there is a certain amount of wandering around and a lot of coffee-drinking, but we get through the agenda in the end. An open meeting is also held at each IETF meeting, so that any member of the IETF can address the IAB.

To understand what the IAB really does in its meetings, it is necessary to know that the detailed work of driving the Internet standards process is done by the IESG. Not only must individual members of the IESG, known as IETF Area Directors, oversee the work of all the working groups in their area, but the IESG as a group must approve all formal standards actions.

This means approving the conversion of Internet Drafts into Proposed Standards, and subsequent steps towards full standardization. Since the last set of reforms of IETF process, in 1992-93, the IAB itself does not have to approve individual standards actions.

The IESG consists of a set of specialists in various technical areas, and IESG positions are filled from the IETF by looking for specialists. In contrast, the IAB members are not appointed as specialists, but rather as generalists with good overview of all aspects of the Internet architecture. In a typical meeting, apart from routine business such as reviewing the IAB action list, we will try to discuss one or two strategic issues in some depth. The intention is to reach conclusions that can be passed on as guidance to the IESG, or turned into published statements, or simply passed directly to the relevant IETF working group.

Examples, Please!

To give some examples, some issues that have been discussed in recent IAB meetings (those between the July and December 1995 IETF meetings inclusive) were:

- The future of Internet addressing

- Architectural principles of the Internet
- Future goals and directions for the IETF
- Management of top level domains in the Domain Name System
- Registration of MIME types
- International character sets
- Charging for addresses
- Tools needed for renumbering

The IAB does not aim to produce polished technical proposals on such topics. Rather, the intention is to stimulate action by the IESG or within the IETF community that will lead to proposals that meet general consensus. In some cases, the IAB does indeed publish Internet Drafts or RFCs but these are in the nature of statements or viewpoints rather than standards proposals. Past experience has shown that standards proposals that have not passed through the fiery experience of peer review by the IETF are unlikely to be generally accepted.

Another type of action that the IAB can trigger is the setting up of a workshop or ad hoc panel, outside the standards process, to develop ideas in a particular area. For example, workshops were held recently on security (RFC 1636) and information infrastructure (RFC 1862).

The IAB can also stimulate the formation of research groups, which is why the IRSG Chair sits in the IAB. These are expected to have a longer existence than panels or workshops, but do not normally produce standards-track documents.

And In Between Meetings?

IAB members try to track the email activity on the main IETF list and on the lists of whichever IETF Working Groups interest them. They can of course intervene as individuals in these discussions whenever they want, but cannot speak in the name of the IAB unless there is a clear consensus.

The IAB Chair, and the nominated IAB Liaison to the IESG, take part in two weekly IESG telephone conferences and track the email activity of the IESG.

While they do not have a vote in formal IESG ballots, they can offer advice on any issue discussed by the IESG and of course refer back to the IAB if necessary. The IAB tends to get involved with IESG discussion at three critical junctures in the life of an IETF working group:

- When a new working group is chartered, the IAB may comment on the draft charter before it is approved.
- When documents reach the last call prior to an IESG ballot, IAB members tend to sit up, pay attention, and stick their oar in.
- When a working group gets into a difficult situation, or tension arises between the WG and the relevant IESG Area Director, IAB members try to help individually or collectively to resolve the situation.

Liaison

The IAB also has a role in external representation and formal liaison. The IETF is far from alone in the world of information technology standards. In a few cases (subcommittees of ISO-IEC/JTC1, ITU-T, The ATM Forum), the IETF has established formal liaisons with other bodies, and the IAB (with the Internet Society) has assisted in the bureaucratic part of this. The real technical liaison of course takes place at WG level. More generally, IAB members find themselves contacted by a wide variety of other organizations in search of information, technical contacts, conference speakers, and the like.

According to its charter (RFC 1601), the IAB has several other jobs:

- The IAB appoints the IETF Chair and all other IESG candidates from a list provided by the IETF Nominating Committee. In recent years this has been an easy job, thanks to the excellent job done by the Nominating Committee.
- The IAB provides oversight of the architecture for the protocols and procedures used by the Internet. The IAB has often discussed exactly what this part of its charter means and how to implement it. The activities described above are the practical realization of this job.
- The IAB provides oversight of the process used to create Internet Standards. The IAB serves as an appeal board for complaints of improper execution of the standards process. Apart from its working relationship with the IESG, IAB members are active in the ongoing review of the current Internet standards process (RFC 1602, under revision). So far there has been only one appeal case, heard in open session by the IAB in April 1995, which resulted in procedural recommendations about the handling of intellectual property rights in the IETF standards process.
- The IAB is responsible for editorial management and publication of the Request for Comments (RFC) document series, and for administration of the various Internet assigned numbers. In fact these responsibilities are fully delegated to the RFC Editor and the IANA respectively, with the IAB available for consultation when needed.
- The IAB acts as a source of advice and guidance to the Board of Trustees and Officers of the Internet Society

concerning technical, architectural, procedural and (where appropriate) policy matters pertaining to the Internet and its enabling technologies. It must be said that this channel has been little used, and a more regular contact between the IAB and the Internet Society is highly desirable.

What We Do Not Do

The IETF is a standards body and the IAB is drawn from the IETF in order to help it achieve its goals of better standardization. For this reason, the IAB has no official role in operational or commercial matters and only a minor role in policy matters. As an example, the IAB could decide to stimulate work on a standard for automatic labeling of email describing how to build nuclear warheads, but could not make policy on whether such messages should be forbidden on international routes.

However, the boundaries of the proper role for the IETF, the IESG, and the IAB are somewhat fuzzy. A real-life example was the request from the Internet Society to the IAB for work to be done on Internet ethics. Although this was hardly standards work, the IETF responded by setting up the working group called "Responsible Use of the Network," which recently finalized RFC 1855, the Netiquette guide.

At the time of writing, the IAB is involved in a discussion about the future management of the Top Level Domains of the DNS. Given that this is an existing responsibility of the IANA, any proposal for change automatically involves the IAB, and raises another example of the ambiguous boundary between standards, policy and operations.

Another fuzzy boundary is "how far up or down do we go?" With the international political drive for information superhighways, the IAB is expecting the Internet to become the infrastructure for the "Information Infrastructure." Does this mean that every information handling protocol must be developed by the IETF? Certainly not! So there will be a boundary between the IETF standards and other information handling standards, and this will not be a completely clear boundary. Similarly, the boundary between IETF standardization and hardware transmission standardization can never be rigid. This is particularly apparent in the case of ATM. The IAB has a role to play in defining the limits of the IETF, closely linked to the question of liaison with other bodies.

In Conclusion

The IAB exists to serve and help the IETF, attempting to strike a balance between action and reaction. IAB members are part-time volunteers (as indeed are IESG members) serving the IETF community with no particular expectation of reward. Of the 12 nominated IAB members at the time of writing, seven are based in the USA, with one each in Australia, Canada, France, the Netherlands and Switzerland. Eight of us work in the computing and telecommunications industry, one in manufacturing industry, two for government-funded research institutes, and one for a university. The days when the IAB could be regarded as a closed body dominated by representatives of the United States Government are long gone. Any IETF member can volunteer for the Nominating Committee, in order to influence the future membership of the IAB.

More Information

The IAB has a Web server with URL: *http://www.isi.edu/iab/*. IAB minutes are also accessible using anonymous FTP from host *ftp.isi.edu*, in directory *pub/IAB*. General information about the IETF is available with the URL: *http://www.ietf.org/home.html*, or from RFC 1718 ("The Tao of the IETF").

The Bookworm

by Peter H. Salus
<peter@pedant.com>

Geoff Collyer wrote me:

> Wandering through Barnes & Noble recently, Mark Moraes and I observed a proliferation of "Uncut" and "Unleashed" books, such as *Linux Unleashed* and *Windows 95 Uncut*, which brought to mind another recent trend. I predict a confluence that will produce such titles as *The World Wide Web Unplugged*, *The Internet Unplugged*, *ISDN Unplugged*, and *Windows NT Unplugged*. You heard it here first.

I wouldn't be surprised. The advent of Java has brought me a flood of largely unripe and poorly roasted beans, with many more to come. I went to my local Barnes & Noble to see how many I was missing ("collect 'em all!") and found I had all but three. I'd reviewed only two; this month I'll mention six more. But I've a press release from Prentice-Hall announcing a series and I know Addison-Wesley's got another series in the works. Some of the current crop of beans is useful, but, unfortunately, most falls within Matthew Teel's categorizations of Java and JavaScript:

> Java: A very crude programming language developed by Sun Microsystems, very much in its infancy, for which development tools available are such that only a masochistic UNIX-head hardened from years of using incredibly deficient tools would use (enjoy?). It is a platform independent compiled bytecode language that uses a native interpreter previously known as Pseudo-Code (and now the savior of the Internet) that produces very poor performance and that prior to the Java revolution has always been a cause for ridicule and disdain. It is driven by an incredible hate and fear of Microsoft and marketing hype [the likes] of which the computer industry has never known, transcending even that of the Redmond giant.
>
> JavaScript: Developed by Netscape Corp. to provide some kind of functionality to an otherwise brain dead software program known as a browser. Developed under another name (who remembers, who cares?) and renamed to ride the crest of the Java wave, it resembles every scripting language known to man, dumped into a can, shaken up, and then dumped out onto the table. Great for making annoying little "scripts" that scroll very jerky strings of text in the browser's status bar so users can squint and try and read what it says.
>
> Summary: Together these technologies will usher in a new age in which the Web will no longer be the domain of mean-lean browsers that fit on a floppy disk and transfer highly compressible ascii code at very high speeds but will belong to bloated, monolithic browsers of 6 MB or more that resemble a cross between a dumb terminal and Windows 3.X and clog the arteries of the Internet with cutesie little applets and animations that look like they're running on my old Gateway 2000 386SX-16 with a whopping 2 MB of RAM.

I can hardly wait.

Best Book

We're just about halfway through the year, but it's clear to me what the best book I've gotten so far is: it's the revision of Leffler, McKusick, Karels, and Quarterman's 4.3 work. Now by McKusick, Bostic, Karels, and Quarterman, the 4.4BSD volume is about 20% bigger. It's terrific! I have lived for nearly seven years with

Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, and John S. Quarterman, **The Design and Implementation of the 4.4BSD Operating System.** Reading, MA: Addison-Wesley, 1996. ISBN 0-201-54979-4. Pp. xxvi+577.

Arnold Robbins, **Effective AWK Programming** [a.k.a. **GNU AWK User's Guide**]. Seattle, WA: SSC, 1996. [Free Software Foundation, 1996]. ISBN 0-916151-88-3 [1-882114-26-4]. Pp. 322.

Larry J. Hughes, Jr., **Actually Useful Internet Security Techniques**. Indianapolis, IN: New Riders, 1995. ISBN 1-56205-508-9. Pp. 378.

Daniel J. Barrett, **Bandits on the Information Superhighway**. Sebastopol, CA: O'Reilly & Associates, 1996.

Daniel P. Friedman and Matthias Felleisen, **The Little Schemer**. 4th edition. Cambridge, MA: MIT Press, 1996. ISBN 0-262-56099-2. Pp. 196.

Daniel P. Friedman and Matthias Felleisen, **The Seasoned Schemer**. Cambridge, MA: MIT Press, 1996. ISBN 0-262-56100-X. Pp. 210.

Terry Winograd, Ed., **Bringing Design to Software**. Reading, MA: Addison-Wesley, 1996. ISBN 0-201-85491-0. Pp. 321.

Adam Freeman and Darrel Ince, **Active Java.** Reading, MA: Addison-Wesley, 1996. ISBN 0-201-40370-6. Pp. 235.

David Flanagan, **Java in a Nutshell**. Sebastopol, CA: Oreilly & Associates, 1996. ISBN 1-56592-183-6. Pp. 438.

Ed Anuff, **Java Sourcebook**. New York: John Wiley & Sons, 1996. ISBN 0-471-14859-8. Pp. 498.

Michael C. Daconta, **Java for C/C++ Programmers**. New York: John Wiley & Sons, 1996. ISBN 0-471-15324-9. Pp. 443+floppy.

Barry Boone, **Java Essentials for C and C++ Programmers**. Reading, MA: Addison-Wesley, 1996. ISBN 0-201-47946-X. Pp. 304.

Neil Bartlett, Alex Leslie, and Steve Simkin, **Java Programming EXplorer**. Scottsdale, AZ: 1996. ISBN 1-883577-81-0. Pp. 848+CD-ROM.

The Bookworm says: "Several folks wrote me as to how/where they could purchase the Plan9 manuals. Rob Pike informs me that Harcourt, Brace is the distributor."

the 4.3 volume near my workstation because it enabled me to understand many of the things I work with or read about. There is no question in my mind that the 4.4BSD treatise will supplant it as my companion. Best of all, it is one of the few books that can actually be read. Kirk, Keith, Mike, and John deserve great praise for their work and their achievement. This is a must for every one of you who is serious about UNIX, no matter what flavor. After all, where did SVR4 get sockets, job control, reliable signals, filesystem interfaces, etc.?

AWK

Brian Kernighan remarked that working on AWK was the toughest thing he had done because there were three people involved. But over the nearly 20 years since Aho and Weinberger joined with Kernighan to work on it, AWK has proven itself a useful, reliable, sophisticated reporting language. Arnold Robbins has written a useful guide to AWK programming that's available from the FSF (as *GNU AWK User's Guide: Effective AWK Programming*) and from SSC (as *Effective AWK Programming: A User's Guide for GNU AWK*). It's a good book. The SSC version has a more attractive cover for an extra $2.

No, no Java yet . . . coffee at the end of the meal.

Useful Security

Larry Hughes has turned out a handy book that lives up to its title: *Actually Useful Internet Security Techniques*. Hughes covers Kerberos (and authentication in general) very well. And his chapters on SATAN and encryption are lucid and comprehensible. This could be said of Barrett's book, too. But I felt that Barrett was too facile, too chummy. The information in Barrett's 200+ pages would go into one of Hughes's chapters. I liked Hughes's sections on network security a good deal, too.

Scheming and Designing

I confess! I was a LISPer. I've never Schemed, but I like Scheme. To celebrate the 20th [!] anniversary of Scheme, we've got a new edition of *The Little Schemer* (a.k.a. *The Little LISPer*) and a new companion *The Seasoned Schemer.* I think they are really fine. Anyone teaching beginning computer science (or just interested in how to introduce the concepts of programming) should consider them seriously.

On the other hand, it's hard for me to understand what *Bringing Design to Software* is trying to do. I'm an admirer of Terry Winograd. I liked his *Language as a Cognitive Process* a lot, over a decade ago. But *Bringing Design* (which grew out of a 1992 ACM workshop) contains more that's trite and stale than that's new. I thought Mitch Kapor's "Manifesto" was just dandy in *Dr. Dobbs* in 1991. It's less great now. Most of the other essays are more design than software. You'll get better stuff on design from books on design. I will admit that here and there you'll find good bits; but it's not really worth the time unless you're trapped on a plane with only the airline's magazine and yesterday's *Dagens Nyheter.* My favorite: Don Norman's "Design as Practiced" – he's the only one who seems to have read Tom Landauer's *The Trouble With Computers.*

And now the coffee . . .

How many Java books balance on the edge of a worktable? I've got six for this month's column.

It may be interesting to look at four of them in terms of their subtitles (two don't have subtitles, a reprehensible failing), for these illuminate what I think are the aims of the authors and their publishers. The subtitles are "Object-Oriented Programming for the World Wide Web," "The Most Complete Guide to Hands-on Java Programming!," "A Desktop Quick Reference for Java Programmers," and "A Complete Guide to Creating Java Applets for the Web." The fifth book is simply called *Java Essentials for C and C++ Programmers*, whereas the sixth is *Java for C/C++ Programmers*. It comes with a PC-compatible floppy. The "Complete Guide . . ." comes with a CD-ROM. The other four are mere text. Clearly, the publishers want to sell to the Webster wannabees, to the constructors of the myriad pages that I never want to have to look at.

Active Java is the smallest of the five, weighing in at a mere 235 beans. Both Freeman and Ince are British, and it may be that their book's terseness and clarity are due to their training. The ordering of topics is exemplary and the exposition very effective. There's little to say about *Java in a Nutshell.* Flanagan has written about X Windows and about Motif for O'Reilly previously. This is a solid, useful handbook. It's worthwhile to look at *Active Java*'s chapter 7 and Flanagan's chapter 19: both on AWT – the Abstract Window Toolkit. Flanagan gives you the hard-core reference material; Freeman and Ince explicate it.

Anuff's *Java Sourcebook* hardly discusses AWT at all: Anuff is so interested in getting his reader to create applets that he barely has time to explain the tools. But then he's interested in HotJava, Sun's Web browser (written in Java). Daconta spends 70 pages on AWT, and they are well-spent. But I liked the preceding 300 beans even more. Daconta compares Java to ANSI C (pp. 25-75) and Java to C++ (pp. 77-143). He's also got a chapter on Java features not in either C or C++ (pp. 245-287). That means that over 50% of the first 300 beans are worthwhile (if not Java, at least Colombian or Kenyan beans). That's a great percentage. Boone's *Java Essentials* is an overly watered-down brew. It contains fewer beans than Daconta's similarly titled book, but far less flavor. Decaf Java Essentials, perhaps? *Java Programming EXplorer* made

me wince. The book is enormous and there's a CD, too. One wonders why. It's too much text about too few beans.

Hooked on Java: Creating Hot Web Sites with Java Applets

by Arthur van Hoff, Sami Shaio, and Orca Starbuck, Addison Wesley, 1996, ISBN 0-201-48837-X, 181 pp. $29.95

Reviewed by George W. Leach
<gwll@gte.com>

Hooked on Java is the first of a series of books that will be forthcoming from Addison Wesley authored by members of the Java implementation team at Sun Microsystems. This book is intended to introduce Java and instruct readers on incorporating Java applets into their Web pages. It is neither a full-blown language tutorial nor a language reference. A CD-ROM accompanies the book; it contains the beta version of the Java Developers Kit (JDK) for Windows 95, Windows NT, and Solaris 2.x; sample applets; Java-enabled HTML pages; and Java source code examples. Since the publication of this book, the JDK has come out of beta, and an updated version can be obtained from *www.javasoft.com*.

The book is organized into six chapters: "Introducing Java and Java Applets," "Java and the Internet," "Applets Explained," "Cool Applets," "Java in Depth," and "Building an Applet." Appendices provide a quick reference to tie applets discussed in the book with the examples provided on the CD-ROM, a list of Web sites related to Java, source code listings for the chapter 6 applets, and a series of man pages for the Java compiler, Java interpreter, and Appletviewer, all of which are on the CD-ROM.

The first two chapters are very brief (approximately 20 pages total) and provide the reader with a quick introduction to Java, applets, HotJava, some history, a high-level functional description of Java, Java's relationship to the Internet, and a bit of crystal ball gazing about Java's future.

Chapter 3 ("Applets Explained") discusses applets, Java-compatible browsers, what applets can do and provides pointers to some URLs for finding new and interesting applets on the net. This chapter also discusses incorporating applets into HTML, passing parameters to an applet, some of the tools available for working with applets, and some operational aspects of using applets.

"Cool Applets" (Chapter 4) provides the reader with a collection of applets, all of which appear on the accompanying CD-ROM. This chapter is by far the largest in the book, with a page count of 51 out of a total of 181. Each applet is described in a format that is very similar to a UNIX man page. A screenshot of the applet running within a Web browser is provided in each description, along with some icons that indicate features supported, such as graphics, audio, animation, mouse directives, etc. For each applet a URL is provided so that the reader can check to see if an applet has been updated by the original author. This is a fun chapter because you can follow along by firing up a Java-compliant Web browser or the Appletviewer that comes on the CD-ROM to play with each example applet that is described.

The next chapter ("Java in Depth") describes some of the syntax of the language. However, it is still a very cursory treatment. Furthermore, the authors assume a knowledge of C. But I believe that the book will be difficult to read without a basic knowledge of C++. There are just too many O-O concepts that need to be explained first. The topics discussed are classes and interfaces, inheritance, exceptions, threads and synchronization, and some of the basics such as data types and flow control constructs.

Chapter 6 ("Building an Applet") builds upon the language elements introduced in the previous chapter to instruct the reader on using the Java language and the Applet API to build applets. The bulk of this chapter discusses the Abstract Window Toolkit (AWT), which is a set of classes for constructing user interface components. There are a number of example applets that implement the standard graphical elements of drawing a line, drawing a rectangle, creating a button on a screen, displaying images (gif files), drawing a string, dealing with layouts, adding controls, and animation. Other topics that are briefly touched on are a description of the execution phases of applets and how to take advantage of them in your own classes, interfacing with the environment, and built-in calls for dealing with images and audio. The chapter concludes with some suggested projects for extending or enhancing the example graphical applets used in this chapter.

My overall impression of this book is that it will be read once and must be used in conjunction with the online documentation (*www.javasoft.com*). Once a programmer becomes comfortable with the element aspects of the language and some of the basics of the classes available with Java, this book will no longer be useful. Also, as more detailed books appear in the marketplace such, as *Java in a Nutshell* (O'Reilly & Associates), this book will become less attractive. The language and class library treatment is rudimentary. Yet this book requires too much prerequisite knowledge about C++ or O-O concepts for it to be considered suitable for C or other procedural language programmers. For nonprogrammers, the introductory material is fine. But the chapters that address the language and library issues require too much technical background to help that audience.

More information on this book, including a sample chapter, is available on the Addison Wesley home page (*<www.aw.com/devpress/java>*). To see what other titles

will be available in the Java Series in the near future, visit <*www.aw.com/cp/javaseries.html*>.

Bandits on the Information Superhighway

By Daniel J. Barrett, O'Reilly & Associates, 1996, ISBN 1-56592-156-9, 246 pp, $17.95

Reviewed by David J. Bianco
<bianco@itribe.net>

I'm a raving paranoiac. No, really. Don't get me wrong; I'm not the type who constantly looks over his shoulder, afraid that *They* are after him. (*They* have much more important things to do. Ever tried debugging an Orbital Mind Control Laser via a 56 Kbps satellite uplink?) It's just that, when it comes to the Internet, I believe in taking every precaution available to me in order to protect myself and my work. That's why, when O'Reilly offered me the opportunity to review their new book, *Bandits on the Information Superhighway*, I jumped at the chance. In keeping with their reputation, O'Reilly has delivered a well-written, informative book that covers the points every Internet user should know about online safety.

First, let me state that this is not a technical book. You won't learn how to send email, read news, or view Web pages. The point of this book is not to show you how to use the Net, but how to use the Net safely. Nor is this book intended to scare you, talking about how low-life cracker scum will break into your computer, steal your credit card numbers, read your email, and pirate your data. Though these possibilities are acknowledged, they are not the focus either. Think of this book as a tutorial on Internet "street smarts." The narrative, supplemented with anecdotes and tips on every page, covers topics such as protecting your privacy online, recognizing when you're being scammed, protecting children from adult information, and what your rights are while you're logged in.

The thing I liked most about this book was the tips and anecdotes included on each page. Though somewhat distracting for the first couple of chapters (it was hard to know whether to read the page text or the sidebars first), I soon got into the rhythm of the book and was able to fully appreciate them. I was also very impressed with the breadth of coverage. Nearly every aspect of my online paranoia was dealt with. In particular, chapter 7, "Pranks, Spams and Time Wasters," was extremely practical. Craig Shergold doesn't need any more get well cards, the FCC wouldn't dream of taxing our modems, and there is no Good Times virus (someone at our company sent a "warning" to us all just last week, so I found this topic particularly timely). I also very much enjoyed the section on resolving conflicts online. The step-by-step instructions on finding postmasters and domain contacts will be invaluable to those having problems with crooked or harassing users. Also worthy of special note is chapter 8, "Strangers, Friends and Lovers" which details how to safely meet people not only online, but also in person for the first time.

I was unable to find anything I really disliked about this book. The few technical simplifications I found appeared to be attempts to gloss over potentially confusing minutia. Given the audience this book is aimed at, that's probably a good thing.

In short, *Bandits on the Information Superhighway* is an excellent source of practical information about keeping yourself safe while pursuing all things Net-related. Online services and ISPs should consider recommending this book to each new user they bring online; users who read this book are substantially less likely to be taken advantage of, tricked, or otherwise scammed online.

Zen and the Art of the Internet, 4th ed.

By Brendan Kehoe, Prentice Hall, 1996, ISBN 0-13-452914-6, 255 pp, softcover, US $23.00

Reviewed by Rick Umali
<rgu@world.std.com>

My first exposure to *Zen and the Art of the Internet* was Brendan's PostScript file, which I downloaded eagerly in early 1992. At that time, his document was the only "overarching" text on contemporary Internet sites and services. It had the benefit of being concise, and its real-world examples whetted my appetite.

This first edition is still online at this URL:
<http://sundance.cso.uiuc.edu/Publications/Other/Zen/zen-1.0_toc.html>

Reading the chapter outline of the first edition gives testimony to how fast the Internet has grown: there is *no* mention of the World Wide Web or gopher in the first edition.

Today, his book is in its fourth edition, and it is still one of the very concise, classic references to all that is out there on the Internet. And, yes, it not only has a whole chapter on the World Wide Web, but it includes a nice HTML reference card, for those aspiring HTML writers.The book itself is not too large (a little over 250 pages). The layout is appealing. There are a number of cartoons and a number of interesting quotes ("Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it." – Samuel Johnson).

I do wish the book employed a different typeface. The original *Zen* PostScript file used a smaller type, and the layout was achieved (I'm guessing) using one of the UNIX text processors, which lent the document a "mature" air. The type-

face made the document seem less "academic," although I'm sure that was the point.

I felt that the listings of Telnet resources, FTP sites, and country code had a little too much white space and could have been more compact.

The index was not as comprehensive or as accurate as I would have liked. For example, "CyberSex" was cited as p. 205, but p. 205 contained part of the appendix on HTML writing. There were a few other of these, and it makes the index less useful.

Despite these minor nits, the book is very solid, and it is still the book to recommend to newbies. Mr. Kehoe attempts to remain OS-neutral, which means that most recipients of this book would benefit from an OS-specific approach to the Internet. However, *Zen* is more like a map. There's a lot of territory to cover, so the map doesn't spend much time exploring modes of transport.

The book contains ten chapters, five of which cover the basic Internet services: email, FTP, USENET News, Telnet, and the World Wide Web. This is an optimal way to "break down" the Internet, because these are the "usual" services available to people connecting to it. As we all know, there are whole books on the Web, News, and email and numerous lengthy documents on Telnet and FTP. Kehoe's book gives the flavor of each of these tools quickly and concisely.

This high-level approach gives the reader the "whole picture" and a launch pad for further study.

Interspersed throughout the text are numerous Net essentials: Yanoff's Internet Services List, the Jargon File, the List of Publicly Accessible Mailing Lists, and netfind. The chapter titled "Finding Out More" mentions RFCs, the "official" thought stuff of the Internet. Along with these Net essentials are Net stories like Robert Morris's "Internet Worm," the "CMU Coke Machine," and Clifford Stoll's "Stalking the Wily Hacker."

Mr. Kehoe can add himself as a Net story and certainly as a Net celebrity.

I do sense some underlying messages. One is vastness. The Internet is huge, and it's only going to get bigger. He mentions this in his afterword: "The Internet changes daily (if not hourly); this guide is simply a snapshot in time, certainly omitting some aspects of the Net." Agreed.

Another message: it's up to us to explore. He is a guide who points to the horizon and says, "There's great stuff there." It's up to us to unearth this stuff. Clearly, his book gives us the tools, but it's up to us to do the exploring. "[E]very user has the potential to be a competent net.citizen," he writes. His appendix on Kids' Sites on the Internet demonstrates what careful exploration of the Internet can lead to.

Finally, he does have a message of caution. In the preface to the second edition, he exhorts: "this territory we are entering can become a fantastic time-sink. Hours can slip by, people can come and go, and you'll be locked in Cyberspace. Remember to do your work!" In the preface to the fourth edition, he suggests, "Choose your pursuits carefully. . . ."

These warnings are not lost on me. The very "bigness" of the Internet can swallow up a person's "life." This topic is in much debate (see Cliff Stoll's *Silicon Snake Oil*, and Steve Talbott's *The Future Does Not Compute*). These warnings remind us that there is a bigger forest out there, and this is a credit to his book.

After perusing the book and getting to a computer to try some of the examples, I am once again in awe of the "vastness" of the Internet. *Zen*, in its fourth incarnation, continues to be the definitive first book for all visitors to the Internet.

Announcement and Call for Participation

USENIX

USELINUX: Linux Applications Development and Deployment Conference

Co-sponsored by Linux International and the USENIX Association

January 6–10, 1997
Anaheim Marriott Hotel
Anaheim, California
Co-located with the USENIX Annual Technical Conference

Tutorials: *January 6-7, 1997*
Technical Sessions: *January 8-9, 1997*
Business Sessions: *January 10, 1997*
Keynote: *Wednesday, January 8, 1997*
Birds-of-a-Feather Sessions: *January 7-9, 1997*
Vendor Exhibits: *January 8-9, 1997*
Reception: *January 8, 1997*

Conference Chair

Michael K. Johnson, *Linux Journal*

Technical Track Committee

Michael K. Johnson, *Chair, Linux Journal*
Mark Bolzern, *WorkGroup Solutions*
Alan Cox, *3Com Remote Access Products*
Jon 'maddog' Hall, Esq., *Digital Equipment Corporation*
Lorrie LeJeune, *O'Reilly and Associates*
Dr. Tom Miller, *North Carolina State University*
Erik Troan, *Red Hat Software*
Dr. Greg Wettstein, *Roger Maris Cancer Center*

Business Track Committee

Jon 'maddog' Hall, Esq., *Chair, Digital Equipment Corporation*
Jonathan Eunice, President, *Founder, Research Director of Illuminata, Inc.*
Michael K. Johnson, *Editor, Linux Journal*
Lorrie LeJeune, *Product Manager of Internet and Linux, O'Reilly and Assoc.*
Bryan Sparks, *President, Caldera, Inc.*
Paul Winbauer, *Director of Technical Programs, Avnet Computing*
Bob Young, *President, Red Hat Software, Inc.*

Suggestions for Topics: *June 1, 1996*
Submissions Due Date: *July 1, 1996*
Materials Due Date: *November 13, 1996*

Overview

The Linux Applications Development and Deployment Conference (USELINUX) is aimed at three primary audiences: application developers porting or developing Linux applications, systems administrators charged with maintaining Linux systems, and business people who wish to bring Linux applications to market. Two technical tracks on January 8th and 9th will include separate components for developers and system administrators. A business track devoted to explaining the dynamics of the Linux marketplace and how to partake of it will take place on January 10.

In addition to the three days of presentations and discussions, there will be two days of tutorials, Birds-of-a-Feather sessions, and Vendor Exhibits.

Tutorials—Monday and Tuesday, January 6-7, 1997

We are actively seeking proposals for half or full day tutorials on practical, technical aspects of using Linux. If you would like to present a tutorial, please contact the USENIX tutorial co-ordinator, Daniel V. Klein.
Phone: 412.421.0285
Fax: 412.421.2332
Email: *dvk@usenix.org*

Technical Track Topics — Wednesday and Thursday, January 8-9, 1997

The technical track will have three components:

- an application developers component focusing on topics helpful in porting and developing Linux applications. These include things like APIs, including both Linux's levels of compliance with industry standard APIs and Linux-specific APIs, and capabilities that give extra functionality and/or convenience.
- a system administrators component to help sysadmins apply their skills to Linux administration, and also demonstrate some of the unique and useful features that can make their lives easier.
- a component for enthusiasts involved in developing the Linux operating system and environment.

Business Track Topics— Friday, January 10, 1997

The business track program will focus on obstacles and challenges in integrating Linux into a business. The target audience are software application developers, hardware vendors, service providers, large in-house application developers, and others who would like to know how to make their business more successful using Linux. This track will concentrate on business issues such as:

- The Linux Market: Who, What, Where, When and Why?

- Application Portfolios: What is available, what can be done?
- Marketing to the Linux Marketplace
- Channels: Retail, Resellers, Distributors, Integrators, OEMs, Service
- Licenses and Licensing: I don't want to give away my application!!!

In addition to the meetings, there will be a compendium of information on CD-ROM for conference attendees. This will include copies of the slides, pointers to resources, white papers, lists of current resellers, user groups, etc.

How to Submit a Proposal

Suggestions for additional topics and areas where the topics can be expanded are welcome by June 1, 1996. The program committee will then prioritize the topics and create the final list of topics by July 1st. The program committee will solicit volunteers to work on the program, and will balance volunteers with program needs by that time.

An expanded electronic version of this announcement with greater detail, as well as the Call for Papers for the USENIX Annual Technical Conference, is available at the USENIX Web site:
http://www.usenix.org.

Proposals for invited talks and panels should be received by July 1, 1996. We welcome submissions of a full paper or an extended abstract. Panel proposals should contain a list of names of potential panelists.

Please send submissions to the program chairs via one of the following methods. All submissions will be acknowledged.

Please send comments, suggestions, and information about volunteering of time for a particular area of expertise (including a small bio of your experience) to the program chair via one of the following methods.

Preferred Method

If you are submitting an idea for the technical track, send email to: *michael@usenix.org.*

For the business program, send your ideas via email to *maddog@usenix.org* with the subject line: MADDOG, ANOTHER GREAT IDEA FOR THE *FABULOUS* USELINUX BUSINESS TRACK

Alternate Method:

Via postal address or fax to:

Michael Johnson or Jon "maddog" Hall
USENIX Association
2560 Ninth Street, Suite 215
Berkeley CA 94710
Fax: 510 . 548 . 5738

Vendor Exhibits

Vendors will demonstrate their products in a relaxed environment where attendees can discuss product features and services.

Vendors are invited to participate in the Vendor Exhibits. This is an excellent opportunity to receive feedback from our technically astute audience. If your company would like to display its products or services, please contact Cynthia Deno.
Phone: 408 . 335 . 9445
Email: *display@usenix.org*

Birds-of-a-Feather Sessions (BOFs)

BOFs are very informal, attendee-organized sessions held in the evenings by attendees interested in a particular topic. They may be scheduled on-site or in advance by contacting the USENIX Conference Office. Send email to *conference@usenix.org* or phone 714 . 588 . 8649.

About the USENIX Association

USENIX is the UNIX and Advanced Computing Systems Technical and Professional Association. Since 1975 the USENIX Association has brought together the community of engineers, system administrators, scientists, and technicians working on the cutting edge of the computing world.

The USENIX conferences have become the essential meeting grounds for the presentation and discussion of the most advanced information on new developments in all aspects of advanced computing systems.

The USENIX Association and its members are dedicated to:

- problem-solving with a practical bias,
- fostering innovation and research that works,
- communicating rapidly the results of both research and innovation,
- providing a neutral forum for the exercise of critical thought and the airing of technical issues.

SAGE, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX.

About Linux International

Linux International (LI) is a new organization dedicated to promoting the development and use of the Linux operating system worldwide. LI promotes Linux in various ways:

- LI has established a "Development Grant Fund" which is used to help Linux developers afford hardware and software that they need to continue development. The money in the fund is primarily given by Linux users eager to contribute to Linux development.
- LI facilitates Linux activities at conferences and trade shows.
- LI's Vendor Assistance Program offers vendors various kinds of assistance in porting their products to Linux.
- LI's Linux Promotion Project disseminates promotional material
- LI's Linux Promotion Project disseminates promotional materials, including press releases, all over the world. What's the point of a world-class operating system if people don't know it exists?

See the LI home page at *http://www.li.org/linux-int/* or send email to *info@li.org* for more information.

Registration Information

The complete program will be available in September 1996. To receive registration information, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 215
Lake Forest, CA 92630
Phone: 714 . 588 . 8649
Fax: 714 . 588 . 9706
URL: *http://www.usenix.org*
Email: *conference@usenix.org*

Or you can send email to our mailserver at *info@usenix.org.* Your message should contain the line: *send catalog* and a catalog will be sent to you.

Second Conference on Object-Oriented Technologies and Systems (COOTS)

June 17-21, 1996
Marriott Eaton Centre
Toronto, Canada

Dave Thomas of Object Technology International will give the keynote address, and there will be an Advanced Topics Workshop.

Tutorial Program
Monday & Tuesday, June 17-18

Java: A Language for Providing Content on the World Wide Web
Jim Waldo, Sun Microsystems Labs and JavaSoft

New ANSI C++ Features
José Lajoie, IBM Canada Laboratory

Introduction to CORBA and CORBA Services
Bruce Martin, SunSoft, Inc.

STL In Action
Graham Glass, ObjectSpace, Inc.

Programming Distributed Components Using Network OLE and C++
Don Box, DevelopMentor

Introduction to the Python Programming Language
Jim Fulton, Consultant

OO Design Patterns for Concurrent, Parallel, and Distributed Systems
Douglas C. Schmidt, Washington University, Missouri

Building Distributed Applications With CORBA and C++
Steve Vinoski, Hewlett-Packard

Pattern-Oriented Software Architecture
Hans Rohnert, Siemens AG

Inter-Domain Management: CORBA, OSI, SNMP
Subrata Mazumdar, Bell Laboratories, Lucent Technologies

Java Applets and the AWT
Nataraj Nagaratnam, Syracuse University

Advanced Modeling and Design for Java Systems
Desmond F. D'Souza and Petter Graff, Icon Computing, Inc.

Technical Program
Wednesday, June 19

Keynote Address:
Experiences on the Road to Object Utopia: An Industrial Research and Development Perspective
Dave Thomas, Object Technology International

Refereed Papers:
Compiler Optimization of C++ Virtual Function Calls
David Bernstein, Yaroslav Fedorov, Sara Porat, Joseph Rodrigue, and Eran Yahav, IBM Haifa Research Lab.

Composing Special Memory Allocators in C++
Keith Loepere, Open Software Foundation

Building Independent Black Box Components in C++
Mark Addesso, Software AG

Interlanguage Object Sharing with SOM
Jennifer Hamilton, IBM

Extending a Traditional OS Using Object-Oriented Techniques
Jose Bernabeu, Vlada Matena, and Yousef Khalidi, Sun Microsystems, Inc.

Object Caching in a CORBA Compliant System
R. Kordale and M. Ahamad, Georgia Tech; M. Devarakonda, IBM T.J. Watson Research

Asynchonrous Notifications Among Distributed Objects
Yeturu Aahlad, Bruce E. Martin, Mod Marathe, and Chung Lee, SunSoft, Inc.

Preliminary Design of ADL/C++ – A Specification Language for C++
Sreenivasa Rao Viswanadha, SUNY Albany

Software Composition with Extended Entity-Relationship Diagrams
Pornsiri Muenchaisri and Toshimi Minoura, Oregon State University

Testing Process Metrics
John McGregor and S. Srinivas, Clemson University

Thursday, June 20

Design Patterns for Dealing with Dual Inheritance Hierarchies in C++
Robert Martin, Object Mentor

The Object Group Design Pattern
Silvano Maffeis, Olsen & Associates, Zurich

Pattern Languages for Handing C++ Resources in an Exception-Safe Way
Harald Mueller, SIEMENS

A Pragmatic Approach to Flexibility
Kai-Uwe Maetzel and Walter Bischofberger, UBILAB

Design and Performance of an Object-Oriented Framework for High-Performance Electronic Medical Imaging
I. Pyarali, T. Harrison, and D. Schmidt, Washington University

Class Relationships and User Extensibility in Solid Geometric Modeling
James R. Miller, University of Kansas

A Distributed Object Model for the Java™ System
Ann Wollrath, Roger Riggs, and Jim Waldo, Sun Microsystems, Inc.

Smart Messages: An Object-Oriented Communication Mechanism
Eshrat Arjomandi, William O'Farrell, Greg Wilson, IBM

Pickling State in the Java™ System
Roger Riggs, Jim Waldo, and Ann Wollrath, Sun Microsystems, Inc.

Highly Concurrent Distributed Knowledge Objects
K.L. Clark and T.I. Wang, Imperial College

Advanced Topics Workshop
Friday, June 21

Distributed Object Computing on the Internet

Look for the full program on *comp.org.usenix* and our WWW site, <*http://www.usenix.org*> . The registration brochure will be mailed in early April. If you have any questions, send email to <*conference@usenix.org*> or call the Conference Office at 714 588 8649.

Fourth Annual Tcl/Tk Workshop

July 10-13, 1996
Monterey, CA

If you are a Tcl/Tk researcher or practitioner, mark your calendar now for this workshop. The preliminary program follows. The full program is available on the USENIX Web site: *http://www.usenix.org* or send email to *conference@usenix.org*.

Tk Interoperability Workshop
Tuesday, July 9

This special one-day, by-invitation workshop is being held by the Tcl team at Sun who hopes to learn from the Tcl community about their experiences in porting Tcl and Tk, particularly the difficulties encountered and issues to be addressed.

The cost is $50 in addition to the Tcl/Tk tutorial and registration fees. If you would like to be invited, please email Jacob Levy: *jyl@eng.sun.com* or Tom Christiansen: *tchrist@mox.perl.com.*

Tutorial Program
Wednesday, July 10

Advanced Topics in Tcl/Tk
George Howlett, AT&T Bell Labs; Ioi Lam, Cornell University; and Michael J. McLennan, AT&T Bell Labs

Introduction to User Interfaces: Methodology, Issues, and Trends
Joseph A. Konstan, University of Minnesota

Writing Cross Platform Scripts with Tcl/Tk
Ray Johnson and Scott Stanton, Sun Microsystems Laboratories

New Features in Tcl 7.5 and Tk 4.1
Brent Welch and Steve Uhler, Sun Microsystems Laboratories

Technical Program
Thursday, July 11th

Managing Complexity in TeamRooms, a Tcl-Based Internet Groupware Application
Mark Roseman, University of Calgary

Agent Tcl: A flexible and secure mobile-agent system
Robert S. Gray, Dartmouth College

TclJava: Portable Extensions
Ken Corey and Scott Stanton, Sun Microsystems Laboratories

A Tk Netscape Plugin Module
Jacob Levy, Sun Microsystems Laboratories,

TclDG – A Tcl Extension for Dynamic Graphs
John Ellson and Stephen North, Lucent Technologies

Backtracking and Constraints in Tcl-BC
Dayton Clark and David M. Arnow, Brooklyn College

QuaSR: A Large-Scale Automated, Distributed Testing Environment
Steven Grady, G. S. Madhusudan, and Marc Sugiyama, Sybase

TclSolver: An Algebraic Constraint Manager for Tcl
Kevin B. Kenny, GE Corporate R&D Center

The NR Newsreader
Jonathan L. Herlocker, University of Minnesota

Tcl/Tk in the Development of User-Extensible Graphical User Interfaces
John M. Skinner, Brookhaven National Laboratory; Richard S. LaBarca, Carnegie Mellon University; and Robert M. Sweet, Brookhaven National Laboratory

Visual Tcl: Building a Distributed MultiPersonality GUI Toolkit for Tcl
Mike Hopkirk, Santa Cruz Operation

Works in Progress

Presentation and Discussion on the Future of Tcl/Tk

Friday, July 12

An On-the-fly Bytecode Compiler for Tcl
Brian T. Lewis, Sun Microsystems Laboratories

Tcl/Tk as an OpenDoc Scripting Part
Jim Ingham , AT&T Bell Laboratories

In Search of the Perfect Mega-widget
Stephen A. Uhler, Sun Microsystems Laboratories

Report from the Tk Portability Workshop
Jacob Levy, Sun Microsystems Labs

SWIG : An Easy to Use Tool for Integrating Scripting Languages with C and C++
David M. Beazley , University of Utah

An Approach to the Automated Wrapping
of a C++ Class Library into Tcl
Ken Martin, General Electric Corporate Research and Development

Tksh: A Tcl Library for KornShell
Jeffrey Korn, Princeton University

Works in Progress

SurfIt! – A WWW Browser
Steve Ball, Cooperative Research Centre for Advanced Computational Systems, Australian National University

Tcl/Tk HTML Tools
Brent Welch and Steve Uhler, Sun Microsystems Laboratories

Programming the Internet with Tcl and Audience1™
Adam Sah, Kevin Brown, and Eric Brewer, University of California, Berkeley

Writing CGI scripts in Tcl
Don Libes, National Institute of Standards and Technology

3:30-5:00: Internet Discussion Panel

Saturday, July 13

Current Tcl/Tk Topics

Lessons from the Neighborhood Viewer: Building Innovative Collaborative Applications in Tcl and Tk
Alex Safonov, Douglas Perrin, Joseph A. Konstan, and John Carlis, University of Minnesota

High Performance Graphic Display With Tcl/Tk
Darren Spruce, European Synchrotron Radiation Facility

Hypertools in Image and Volume Visualization
Pierre-Louis Bossart, Lawrence Livermore National Laboratory

A Clinical Neurophysiology
Information System based on Tcl/Tk
Martin Andrews and Richard C. Burgess, The Cleveland Clinic Foundation

Sixth USENIX Security Symposium

Focusing on Applications of Cryptography

July 22-25, 1996
Fairmont Hotel, San Jose, CA

If you are responsible for the safety and security of your employer's computer network, you will want to attend this Symposium. Here is the preliminary program. For complete information, visit the USENIX Web site: *http://www.usenix.org* or send email to: *conference@usenix.org*.

Tutorial Program
Monday & Tuesday, July 22-23

Security on the World Wide Web
Daniel Geer, OpenMarket, Inc., and Jon Rochlis, BBN Planet

Security for Software Developers:
How to Write Code that Withstands Hostile Environments
Marcus J. Ranum, V-ONE Corporation

UNIX Security Tools: Use and Comparison
Matt Bishop, University of California, Davis

Internet Security for System and Network Administrators
Ed DeHart, Computer Emergency Response Team (CERT)

Keys to Successfully Implementing Cryptography
Bruce Schneier, Counterpane Systems

Refereed Papers
Wednesday & Thursday, July 24-25

A Secure Environment for Untrusted Helper Applications
Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer, University of California, Berkeley

A DNS Filter and Switch for Packet-filtering Gateways
Bill Cheswick, Lucent Technologies and Steven M. Bellovin, AT&T Research

Confining Root Programs with Domain
and Type Enforcement
Kenneth M. Walker, Daniel F. Sterne, M. Lee Badger, Michael J. Petkac, David L. Sherman, and Karen A. Oostendorp, Trusted Information Systems

SSH – Secure Login Connections Over the Internet
Tatu Ylonen, Helsinki University of Technology

Dual-workfactor Encrypted Key Exchange: Efficiently
Preventing Password Chaining and Dictionary Attacks
Barry Jaspan, Independent Consultant

Security Mechanism Independence in ONC RPC
Mike Eisler, Roland J. Schemers, and Raj Srinivasan, Sun Microsystems

Secure Deletion of Data from Magnetic
and Solid-State Memory
Peter Gutmann, University of Auckland

Establishing Identity Without Certification Authorities
Carl Ellison, Cybercash, Inc.

A Revocable Backup System
Dan Boneh and Richard J Lipton, Princeton University

Achieving Atomicity in Electronic Commerce
and Its Impact on Communication Efficiency
Jiawen Su and J.D. Tygar, Carnegie Mellon University

Kerberos on Wall Street
Isaac Hollander, P. Rajaram, and Constantin Tanno, Morgan Stanley, Inc.

A Framework for Building an Electronic Currency System
Lei Tang, Carnegie Mellon University

Chrg-http: A Tool for Micropayments
on the World Wide Web
Lei Tang, Carnegie Mellon University; Steve Low and Nicholas Maxemchuck, AT&T Research

Building Systems that Flexibly
Download Executable Content
Trent Jaeger and Atul Prakash, University of Michigan; Aviel D. Rubin, Bellcore

Enabling Secure Collaboration Over the Internet
Li Gong, SRI International

Public Key Distribution with Secure DNS
James M. Galvin, EIT/VeriFone

Compliance Defects in Public-Key Cryptography
Don Davis, Independent Consultant

Texas A&M University Anarchistic Key Authorization
(AKA)
David Safford, David Hess, and Douglas Schales, Texas A & M University

Murphy's Law and Computer Security
Wietse Venema, Eindhoven University of Technology

NetKuang – A Multi-Host Configuration Vulnerability Checker
Dan Zerkle and Karl Levitt, University of California, Davis

Problem Areas for the IP Security Protocols
Steven M Bellovin, AT&T Research

UniForum Panel Sessions
Wednesday, July 24

Security and Privacy Issues
Chair: Peter Neumann, Principal Scientist, Computer Science Laboratory, Stanford Research Institute

Panelists: Mary Connors, Computer Professionals for Social Responsibility; Jose Martinez, Sausalito Associates International; and Gio Wederhold, Stanford University

Electronic Commerce
Chair: Rik Farrow, Consultant and Author

Panelists: Fred Avolio, Trusted Information Systems; Dan Geer, Open Market; and Bruce Schneier, Counterpane Systems

Cryptography and the Law
Chair: Daniel Appelman, Attorney , Heller Ehrman White & McAuliffe

Panelists to be announced.

Cryptographic Infrastructure
Chair: Fred Avolio, Principal Analyst and Product Manager, Trusted Information Systems (TIS)

Panelists: Peter Dinsmore, Trusted Information Systems, Carl Ellison, Cybercash, Inc., Constantin Tanno, Morgan Stanley

Invited Talks
Thursday, July 25

Just Another Convicted Perl Hacker
Randall Schwartz, Stonehenge Consulting Services

Using Technical Means to Protect Individual Privacy: The C2.Net Privacy Model
Sameer Parekh, Community ConneXion

Firewalls: Are They Being Used Right? Are They Cost Effective?
Marcus Ranum, V-ONE Corporation

PGP Library
Derek Atkins, Sun Microsystems

Vendor Exhibits
Wednesday
12 noon - 2:00 pm & 3 pm - 7 pm

For the first time, the USENIX Association will have an informal, table-top Vendor Exhibits at the Security Symposium. Space is very limited, so if your company wants to participate, please contact Cynthia Deno at 408 335 9445 or send email to *cynthia@usenix.org*.

Participating Vendors:
- Computer Security Institute
- Internet Security Systems, Inc.
- Memco Software, Inc.
- Platinum Technology, Inc.
- SecureWare

Terminal Room

PPP and terminal access to the Internet along with dial-out access will be provided in the Terminal Room. The Terminal Room will be open Monday morning through Thursday afternoon. Some access methods (currently undetermined) will be available. Attendees can request Kerberos IDs and/or IP addresses for portables. Terminal Room volunteers will receive a complimentary technical sessions registration. Look for details posted in *comp.org.usenix*.

Student Stipends Available

The USENIX students stipend program covers travel, living expenses, and registration fees to enable full-time students to attend USENIX meetings. Detailed information about applying for a stipend is available at the USENIX Web site, *http://www.usenix.org*, by reading *comp.org.usenix*, or sending email to *students@usenix.org*.

PGP Key Signing Service

USENIX members will be able to sign up for the PGP Key Signing Service that allows messages to be exchanged across public networks while protecting the privacy of the message contents and guaranteeing the authenticity of the sender.

Fifth International Workshop on Object-Orientation in Operating Systems: IWOOOS '96

October 27-28, 1996 – Seattle, WA

Sponsored by the IEEE Technical Committee on Operating Systems and Application Environments (TCOS) and the USENIX Association

The fifth International Workshop on Object-Orientation in Operating Systems will bring together researchers and practitioners who are interested in object-oriented approaches to operating systems design, development, and application support. The purpose of the workshop is to provide an informal format and atmosphere in which ideas and current work can be presented and discussed at length. The workshop is designed to encourage the full participation of each attendee: both presenters and participants will be active contributors throughout the workshop.

This year's workshop will be held in Seattle, Washington, just prior to the Second Symposium on Operating Systems Design and Implementation which will be held in Seattle, Washington from October 28-31. We hope that the conjunction of the two events will foster cross-fertilization between related research communities.

The focus of the workshop is on how to effectively use objects inside operating systems and how to provide system support for object-oriented applications in a variety of application domains.

Subjects of particular interest include:

- Using objects to make operating systems customizable, extensible and adaptable
- Design patterns in operating systems
- Objects on WWW and their OS support
- Persistent objects and their OS support
- Mobile Objects and their OS support

The workshop is structured to encourage the submission of explorative work in the form of position papers. Position papers should be a maximum of 2500 words and should present initial work, new ideas, or a strong position statement.

Attendance will be by invitation only. To be invited, an attendee must submit a position paper. All submissions will be reviewed. All accepted papers will be published in a proceedings. The official language for the conference will be English.

Organizing Committee

Workshop Chair: Andrew Black, *Oregon Graduate Institute*
Program Chair: Nayeem Islam, *IBM T. J. Watson Research Center*
Local Arrangement Co-Chairs: Michael Jone, *Microsoft* and Crispin Cowan, *Oregon Graduate Institute*
Publicity Chair: Douglas Schmidt, *Washington University, St. Louis*
Publication Chair: Luis-Felipe Cabrera, *IBM*
Finance Chair: David Cohn, *University of Notre Dame*

Program Committee

Mustaque Ahamad, *Georgia Institute of Technology*
Henri Bal, *Vrieje University*
Gary Lindstrom, *University of Utah*
Eric Manning, *University of Victoria*
Satoshi Matsuoka, *University of Tokyo*
Gregor Kiczales, *Xerox Parc*
Sacha Krakowiak, *IMAG, France*
Jim Purtilo, *University of Maryland*
John Rosenberg, *University of Sydney*
Margo Seltzer, *Harvard University*
Santosh Shrivastava, *University of Newcastle-upon-Tyne*
Mario Tokoro, *Keio University*

Important Dates

Position Papers: July 1, 1996
Invitations issued: July 30, 1996
Camera-ready copy due: September 1, 1996

Submission Instructions

Each submission should have a principal author, to whom all messages will be sent; please provide email and postal addresses as well as telephone and fax numbers. A notice will be sent to the principal author upon receipt of every paper.

Electronic submission via email is strongly encouraged. Please send your paper to the program chair at *nayeem@watson.ibm.com*

Submissions are required to be in HTML, ASCII, or PostScript (uuencoded). Electronic submissions will be ACK'ed within a day or so of receipt. If the submission could not be successfully printed out on paper then the program chair will attempt to confer with the sender via email about what to do as an alternative submission means. Note: if you do not receive any acknowledgment message at all within a period of several days then the submission message may have gotten lost in transit and you should send a short email message to the program chair to alert him to the difficulty.

Announcement and Call for Papers USENIX

Second USENIX Workshop on Electronic Commerce

November 18–20, 1996
Claremont Hotel & Resort, Oakland, CA

Sponsored by the USENIX Association

Important Dates

Extended abstracts due: *July 16, 1996*
Notification to authors: *August 5, 1996*
Camera-ready final papers due: *October 7, 1996*

Preliminary Program Committee

Ross Anderson, *Cambridge University*
Nathaniel Borenstein, *First Virtual Holdings, Inc.*
Stefan Brands, *CWI*
Dan Geer, *Open Market, Inc.*
Mark Manasse. *DEC Systems Research Center*
Clifford Neuman, *University of Southern California*
Doug Tygar, *Carnegie Mellon University,* Program Chair
Hal Varian, *University of California, Berkeley*
Bennet Yee, *University of California, San Diego*
Others to be determined

Overview

The Second USENIX Workshop on Electronic Commerce will provide a major opportunity for researchers, experimenters, and practitioners in this rapidly self-defining field to exchange ideas and present results of their work. This meeting will set the technical agenda for work in the area of Electronic Commerce by examining urgent questions, discovering directions in which answers might be pursued, and revealing cross-connections that otherwise might go unnoticed.

Tutorials

The Workshop will begin with a day of tutorials. The tutorial program will offer a selection of tutorials from among several tracks on such topics as cryptography and security.

Workshop Topics

Two days of technical sessions will follow the tutorials. Submissions are welcome for technical and position paper presentations, reports of work-in-progress, technology debates, and identification of new open problems. Birds-of-a-Feather sessions in the evenings and a keynote speaker will round out the program.

We seek papers that will address a wide range of issues and ongoing developments, including, but not limited to:

- Advertising
- Anonymous transactions
- Auditability
- Business issues
- Copy protection
- Credit/Debit/Cash models
- Cryptographic security
- Customer service
- Digital money
- E-mail enabled business
- EDI
- Electronic libraries
- Electronic wallets
- Exception handling
- Hardware-enabled commerce
- Identity verification
- Internet/WWW integration
- Key management
- Legal and policy issues
- Micro-transactions
- Negotiations
- Privacy
- Proposed systems
- Protocols
- Reliability
- Reports on existing systems
- Rights management
- Service guarantees
- Services vs digital goods
- Settlement
- Smart-cards

Questions regarding a topic's relevance to the workshop may be addressed to the program chair via electronic mail to *tygar@cs.cmu.edu.* Proceedings of the workshop will be published by USENIX and will be provided free to technical session attendees; additional copies will be available for purchase from USENIX.

What To Submit

Technical paper submissions and proposals for panels must be received by July 16, 1996. We welcome submissions of the following type:

Refereed Papers—Full papers or extended abstracts should be 5 to 20 pages, not counting references and figures.

Panel proposals—Proposals should be 3 to 7 pages, together with a list of names of potential panelists. If accepted, the proposer must secure the participation of panelists, and the proposer will be asked to prepare a 3 to 7 page summary of panel issues for inclusion in the Proceedings. This summary can include position statements by panel participants.

Please accompany each submission by a cover letter stating the paper title and authors along with the name of the person who will act as the contact to the program committee. Please include a surface mail address, daytime and evening phone number, and, if available, an email address and fax number for the contact person. If all of the authors are students, please indicate that in the cover letter for award consideration (see "Awards" below).

USENIX workshops, like most conferences and journals, require that papers not be submitted simultaneously to more than one conference or publication and that submitted papers not be previously or subsequently published elsewhere. Submissions accompanied by "non-disclosure agreement" forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Awards

The program committee will offer awards of $500 for the best paper and the best student paper.

Where To Submit

Please send submissions to the program committee via one of the following methods. All submissions will be acknowledged.

Preferred Method: email (Postscript)

Authors should ensure that their papers will print on a broad range of postscript printers and submit in sufficient time to allow us to contact the author about alternative delivery mechanisms in the event of network failure.

Alternate Method: 10 copies, via postal delivery to

Doug Tygar (program chair)
Computer Science Dept, CMU
5000 Forbes Ave
Pittsburgh, PA 15213-3891
Email: *tygar@cs.cmu.edu*
Fax: 412.268.5576

If you have questions on the format of submissions or about the workshop, please telephone the USENIX Association office at 510.528.8649, or email to *ecauthors@usenix.org* or the program chair *tygar@cs.cmu.edu.*

An electronic version of this Call for Papers is available at WWW URL:

http://www.usenix.org

Registration Information

Materials containing all details of the technical and tutorial programs, registration fees and forms and hotel information will be available in September 1996. If you wish to receive the registration materials, please contact USENIX at:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630
Phone: 714.588.8649
Fax: 714.588.9706
Email: *conference@usenix.org*
URL: http://www.usenix.org

Or you can send email to our mailserver at *info@usenix.org.* Your message should contain the line: send catalog. A catalog will be returned to you.

Announcement and Call for Participation USENIX

USENIX Annual Technical Conference

Pre-Announcement and Call for Papers and Presenters

January 6-10, 1997
Anaheim Marriott Hotel
Anaheim, California

Program Committee

John T. Kohl, Program Chair, *Atria Software*
Matt Blaze, *AT&T Research*
Bill Bolosky, *Microsoft Research*
Nathaniel Borenstein, *First Virtual Holdings*
Charlie Briggs, *Digital Equipment Corporation*
Clem Cole, *Locus Computing*
Fred Douglis, *AT&T Research*
Rob Gingell, *Sun Microsystems*
Mike Karels, *Berkeley Software Design*
Peg Schafer, *Harvard University*
John Schimmel, *Silicon Graphics*
Carl Staelin, *Hewlett-Packard Laboratories*

Invited Talks Co-ordinators

Mary Baker, *Stanford University*
Berry Kercheval, *Xerox PARC*

Due Dates for Refereed Paper Submissions

Manuscripts Due: *June 18, 1996*
Notification to Authors: *August 7, 1996*
Camera-ready Final Papers Due: *November 13, 1996*

Conference Schedule Overview

Tutorials: *January 6–7, 1997*
Technical Sessions and Invited Talks: *January 8–10, 1997*
Birds-of-a-Feather Sessions: *January 7–9, 1997*
Vendor Display: *January 8–9, 1997*
USENIX Reception: *January 8, 1997*

Conference Overview

The conference technical sessions include one track of refereed papers selected by the Program Committee. The refereed papers are published in the Conference Proceedings which are provided to all registered technical session attendees.

There is also a parallel track of Invited Talks. These survey-style sessions given by experts range over a variety of interesting and timely topics. Submitted Notes from the Invited Talks are published and distributed to registered technical sessions attendees.

Two full days of tutorials precede the technical sessions with practical tutorials on timely topics.

Other highlights of the conference include a work-in-progress session, which provides a forum for short informal technical presentations; the evening birds-of-a-feather sessions which provide very informal gatherings on particular topics; the Guru is IN sessions, informal discussions where noted experts from the USENIX community will answer technical questions; and vendor exhibits, which provide the opportunity for no-nonsense evaluation of products and services.

Refereed Papers

The emphasis for the 1997 USENIX Technical Conference is on advanced systems' uses in the global computing environment. How do we build computing systems which fulfill current needs yet can grow to handle the future demands? What techniques and technologies can we use to satisfy a large, growing, and changing computing appetite? How do we support new computing styles with advanced computing systems? How do we protect the systems we build from failures or abuses?

The Program Committee seeks original and innovative full papers about the applications, architecture, implementation, and performance of modern computing systems. Some particularly interesting related topics are:

- Scaling the advanced system: down to laptops, palmtops, embedded systems; up to large file systems and memories, mass storage, faster networks, new protocols
- Mobile systems: network connectivity, system support, application design
- Tasks/roles where advanced systems shine or fall short
- Practical network security, privacy, and cryptography
- Electronic commerce, internetworking
- Multi-media challenges, solutions, and innovations
- Interoperation/standards: tools, techniques, and experience connecting with other computing systems

This list is by no means exhaustive; you are encouraged to submit papers on other advanced system related topics. As at all USENIX conferences, papers that analyze problem areas and draw important conclusions from practical experience are especially welcome.

How to Submit a Refereed Paper

It is imperative that you contact the USENIX Association office to receive detailed guidelines and suggestions for submitting a quality paper to the refereed track of the technical sessions. Please send email to

usenix97authors@usenix.org
or telephone 510 . 528 . 8649.

In addition, specific questions about submissions to the USENIX 1997 Technical Conference may be sent to the program chairman via email at *kohl@usenix.org*.

The program committee will review full papers this year (rather than extended abstracts as in the past). Papers should be 8 to 12 single-spaced 8.5" × 11" pages (about 4000–6000 words), not counting figures and references. Papers longer than 12 pages will be discarded without review.

Include references to establish that you are familiar with prior work and how it relates to your work. Where possible/applicable, provide detailed performance data to establish that you have a working implementation and measurement tools. A good paper will demonstrate that the authors:

- are attacking a significant problem,
- are familiar with the current and past literature about the problem,
- have devised an original or clever solution,
- if appropriate, have implemented the solution and characterized its performance,
- have drawn appropriate conclusions about what they have learned.

Note that the USENIX Technical Conference, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors will be notified by August 7, 1996. Some accepted papers will be shepherded by a program committee member through an editorial review process prior to publication in the conference proceedings.

Where to Send Submissions

Please send one copy of your manuscript to the program chairman via one of the following methods. All submissions will be acknowledged.

Preferred method: email (PostScript or ASCII) to: *usenix97papers@usenix.org*

If you have a MIME-capable mail system, you are encouraged to include your PostScript manuscript as Content-Type: application/postscript, Content-Transfer-Encoding: base64. Important: For PostScript submissions, use only standard PostScript fonts, and format your paper for US Letter (8.5 × 11 inches) paper. If your paper will not print properly, your submission will be returned. You should attach the cover letter (see below) as a separate MIME enclosure.

Alternate method: Postal Delivery to:

John Kohl
Atria Software
20 Maguire Road
Lexington, MA USA 02173
Phone: 617 . 676 . 2641

The authors must also submit the following information (for administrative handling) via email to *usenix97papers@usenix.org*

1. The title of the manuscript and the names of the authors. (Note: the program committee does not review papers blindly; the authors' names and affiliations will be known to the reviewers).
2. The name of one author who will serve as a contact, an email address, day and evening phone numbers, postal mail address, and fax.
3. An indication of which, if any, of the listed authors are full-time students.
4. A short abstract of the paper (100–200 words) (this can be the same as the paper's abstract).

Cash Prizes

Cash prizes will be awarded for the best paper at the conference and the best student paper.

Invited Talks

An Invited Talks track complements the Refereed Paper track. These talks by invited experts provide introductory and advanced information about a variety of topics such as using standard UNIX tools, tackling system administration difficulties, or employing specialized applications. Submitted Notes from the Invited Talks are published and distributed free to technical sessions attendees. This track also includes panel presentations and selections from the best presentations offered at 1996 USENIX conferences and symposia.

Suggestions/Proposals Wanted

The Invited Talks coordinators welcome suggestions for topics and request proposals for particular talks. In your proposal, state the main focus, include a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to

ITusenix@usenix.org.

Tutorials

On Monday and Tuesday, you may attend intensive, immediately practical tutorials on topics essential to the use, development, and administration of UNIX and UNIX-like operating systems, windowing systems, networks, advanced programming languages and related technologies. The USENIX Association's well-respected program offers introductory and advanced tutorials, presented by skilled instructors who are hands-on experts in their topic areas. USENIX will offer two full days of tutorials covering topics such as:

- System and network administration
- System and network security
- Java
- Distributed computing
- Kernel internals: SVR4, BSD, Windows NT
- Systems programming tools and program development
- Portability and interoperability
- Client-server application design and development
- Sendmail, DNS, and other networking issues
- GUI technologies and builders
- WWW technologies

Proposals Wanted

To provide the best possible tutorial slate, USENIX constantly solicits proposals for new tutorials. If you are interested in presenting a tutorial, contact the Tutorial Coordinator:

Daniel V. Klein
Phone: 412 . 421 . 2332
Email: *dvk@usenix.org*

Work-in-Progress Reports (Whips)

Bob Gray, *U S WEST Advanced Technologies*
WIPS Co-ordinator

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The Work-in-Progress reports, scheduled during the technical sessions, introduce interesting new or ongoing work. The USENIX audience provides valuable discussion and feedback. We are particularly interested in presenting student work. To schedule your report, send email to *wips97@usenix.org.*

Birds-of-a-Feather Sessions (BOFs)

The popular evening Birds-of-a-Feather sessions are very informal attendee-organized gatherings of persons interested in a particular topic. BOFs often feature presentations or demonstrations followed by discussion, announcements, and the sharing of strategies. BOFs are offered Tuesday, Wednesday, and Thursday evenings of the conference. They may be scheduled on-site at the conference or in advance by contacting the USENIX Conference Office by phone at 714 . 588 . 8649 or via email to *conference@usenix.org.*

Vendor Exhibits

Vendors will demonstrate the technical innovations which distinguish their products. In this relaxed environment, attendees can discuss the product features and services on display.

Vendors: This is an exceptional opportunity to receive feedback from our technically astute attendees. If your company would like to display its products and services, please contact:

Zanna Knight
USENIX Association
Telephone: 510 . 528 . 8649
Fax 510 . 548 . 5738
Email: *display@usenix.org*

Conference Program and Registration Information

Special Hotel Rates

The Anaheim Marriott Hotel, adjacent to Disneyland, is headquarters for the USENIX 1997 Technical Conference and will be the location for all conference activities. The Anaheim Marriott will be offering special room rates to attendees.

Registration Materials

Materials containing all details of the technical sessions, tutorial program, conference registration, hotel and airfare discounts, and reservation information will be available mid-September, 1996.

If you wish to receive the registration materials, please contact:

USENIX Conference Office
22672 Lambert St., Suite 613
Lake Forest, CA USA 92630
Phone: 714 . 588 . 8649
Fax: 714 . 588 . 9706
Email: *conference@usenix.org*

About The USENIX Association

Since 1975, the USENIX Association has provided a forum where the community of engineers, scientists, and technicians working on the cutting edge of the computing world come together to communicate the results of innovation and research in UNIX and modern open systems. USENIX is well known for its technical conferences, tutorial programs, and the wide variety of publications it has sponsored over the years. USENIX is the original, not-for-profit membership organization for individuals and institutions interested in UNIX and related technologies. Evolving with technology, USENIX has broadened its activities to include open systems and the globally interconnected and interoperable computing environment.

The USENIX Association and its members are dedicated to:

- problem-solving with a practical bias,
- fostering innovation and research that works,
- rapidly communicating the results of both research and innovation, and
- providing a neutral forum for the exercise of critical thought and the airing of technical issues.

SAGE, the System Administrators Guild, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession.

For more information about USENIX and SAGE, our publications, or events, please visit our Web site. The URL is *http://www.usenix.org.* Or, send email to our mailserver at *info@usenix.org.* Your message should contain the line: *send catalog.* A catalog will be returned to you.

A UNIQUE OFFER ON THE BEST IN UNIX FOR USENIX MEMBERS

20% DISCOUNT FROM McGRAW-HILL

☐ **THE INTERNET GUIDE FOR NEW USERS**
D. Dern
hardcover, 016510-6, $40.00,
MEMBER PRICE **$32.00**
paperback, 016511-4, $27.95,
MEMBER PRICE **$22.36**

☐ **INTERNET FOR EVERYONE**
R. Wiggins
hardcover, 067018-8, $29.95,
MEMBER PRICE **$23.96**
paperback, 067019-6, $45.00,
MEMBER PRICE **$36.00**

☐ **THE ESSENTIAL INTERNET INFORMATION GUIDE**
J. Manger
707905-1, paperback, $27.95,
MEMBER PRICE **$22.36**

☐ **THE INFORMATION BROKERS HANDBOOK, Second Edition**
S. Rugge
911878-x, paperback, $34.95,
MEMBER PRICE **$27.96**
Available December 1994

☐ **SAA AND UNIX: IBM's Open System Strategy**
M. Killen
034607-0, $40.00,
MEMBER PRICE **$32.00**

☐ **A STUDENT'S GUIDE TO UNIX**
H. Hahn
025511-3, paperback, $28.00,
MEMBER PRICE **$22.40**

☐ **UNIX DEVELOPER'S TOOL KIT**
K. Leininger
911836-4, $65.00,
MEMBER PRICE **$52.00**

☐ **UNIX SECURITY: A Practical Tutorial**
N. Arnold
002560-6, $24.95,
MEMBER PRICE **$19.96**

☐ **THE UNIX AUDIT: Using UNIX to Audit UNIX**
M. Grottola
025127-4, $32.95,
MEMBER PRICE **$26.36**

☐ **UNIX: A Database Approach**
S. Das
015745-6, $29.95,
MEMBER PRICE **$23.96**
Available November 1994

I am a member of USENIX Association. Please send me the books I have indicated at the member special rate. I have added $3.00 postage and handling for the first book ordered, $1.00 for each additional book, plus my local sales tax.

Check or money order is enclosed—payable to McGraw-Hill, Inc.
Charge my ❑ Visa ❑ Mastercard ❑ Discover ❑ Amex

Account #_____

Expiration Date_____

Signature_____

USENIX Membership # _____

Bill & Ship To:

Name_____

Street_____

City, State, Zip_____

Daytime Phone #_____

03US002

Send or Fax Orders to:

McGraw-Hill, Inc.
Attn: Rosa Perez
11 West 19th Street—4th Floor
New York, New York 10011
Fax: 212-337-4092

If I am not completely satisfied, I will return the book(s) within 15 days for a full refund or credit. Satisfaction unconditionally guaranteed. Prices subject to change without notics. We can only accept orders within the continental USA.

NEW FROM UNIFORUM

Setting Up a Home Page on the World Wide Web

Setting Up a Home Page on the World Wide Web is the new best-seller from UniForum. Written by Michael Harrington, author of *Establishing a World Wide Web Server*, this new guide comes along at just the right time as thousands of computer users at home and on the job want to get on the Web.

Setting Up a Home Page on the World Wide Web shows you how to get started, how Home Pages are built, what makes a Home Page effective and how to get maximum use from your Home Page.

Organized in a step-by-step way, the book makes an ideal text for instructors or a self-teaching guide. Clearly written with numerous illustrations, ***Setting Up a Home Page on the World Wide Web*** is the resource you need to take full advantage of the fastest growing application of the Internet.

Setting Up a Home Page on the World Wide Web is in stock and available for immediate shipping. See the chart below for individual and multiple copy prices. UniForum members always enjoy a discount! Call UniForum today for more information and to place your order.

Note to College and University instructors:
Complimentary desk copies of Setting up a Home Page on the World Wide Web, *and* Establishing a World Wide Web Server, *are available upon request. Please send a letter on official school stationery stating description of prospective course use to: Publisher, UniForum, 2901 Tasman Drive, Suite 205, Santa Clara, CA 95054.*

***Setting Up a Home Page on the World Wide Web* covers what you really need to know:**

- **A general introduction to the Internet and World Wide Web**
- **The major components of a Home Page**
- **Step-by-step instructions on building a Home Page**
- **Different types and uses of Home Pages**
- **Home Page security and tools for easier use**
- **The technical components of a Home Page including HTML, URL, Image Maps, Forms and Common Gateway Interfaces**
- **Five appendices covering identifiers, elements, Netscape extensions and Web Browsers**

All General Members of UniForum will receive a copy of ***Setting Up a Home Page on the World Wide Web*** *as part of their membership benefits.*

Copies are available to all UniForum General and Trial Members at substantial discounts:

QUANTITY	MEMBER PRICE	TRIAL MEMBER PRICE
1-10 copies	$10.00	$25.00
11-49 copies	$ 8.00	$20.00
50 copies or more	$ 7.00	$17.00

Plus shipping and handling

Shipping & Handling Charges	US/Canada/Mexico	International
If your mechandise total is...	Add...	Add...
UP to $25.00	$5.00	$7.50
$25.01 to 50.00	7.00	9.50
$50.01 to 75.00	9.00	11.50
$75.01 or more	11.00	13.50

To order, please call UniForum today at **1-800-255-5620**, or **(408) 986-8840**. Have your Visa, MasterCard, Discover or American Express card ready for fastest order processing. Orders may be sent to UniForum. Please send checks or money orders (US dollars only; California residents please add applicable sales tax) to:

UniForum
Setting Up a Home Page
2901 Tasman Drive, Suite 205
Santa Clara, CA 95054-1100

Credit card orders may also be faxed to us at (408) 986-1645

9603202

UniForum™
The International Association of Open Systems Professionals

UniForum '97

The Official Conference and Exposition for Open Systems Professionals

Call for Proposals and Presenters

Tutorials • Workshops • Seminars • Track Sessions • BOFs
March 10-14, 1997 • Moscone Convention Center • San Francisco

The UniForum Association invites presenters to submit proposals for its annual conference in the following general areas of Unix and open technologies:

- **Operating Systems - Hardware and Software**
- **The Internet, the Intranet, the World Wide Web**
- **Application Development and Tools**
- **Open Network Computing**
- **Client/Server, Middleware and Legacy System Migration**
- **Data Base, Data Mining, Data Warehousing Implementation Strategies**
- **PC and Unix Integration**
- **Systems and Network Administration & Management**
- **Computer Telephony Integration**
- **Security**
- **Emerging and Advanced Technologies**

One and Two-day Conference Workshops, Tutorials and Seminars will take place March 10-11, while Track Sessions and BOFs will run March 12-14.

About the UniForum '97 Conference

The UniForum Conference is the largest educational event of its kind for the practitioner in the enterprise who is working with, or plans to work with, Unix systems and open technologies. The Conference attendee will have a computer systems background, but not necessarily Unix systems experience. The attendee will be looking for substantive, practical information that can be applied on the job or used for strategic planning and/or procurement.

Proposals for UniForum '97 Conference sessions should target this audience. Proposals that are case study oriented and which call for participative feedback are preferred.

What Your Proposal Should Include

Your Proposal should identify the type of session you are interested in giving: multi-day or single-day Workshop, Tutorial or Seminar; Track Session or BOF.

Your Proposal should include a paragraph description about the technology to be discussed, its open technology connection, and a problem/solution case statement that speaks to how the technology works in the enterprise. A target audience should be named with pre-requisites, if any. A brief presenter's background should give pertinent information on expertise and conference experience.

How to Submit Proposals

The preferred method is to send your Proposal via email (ASCII) to: conference97@uniforum.org

Or by mail to: Claudia Marshall, Conference Coordinator, UniForum, 2901 Tasman Drive, Suite 205, Santa Clara, CA 95054. Your submission will be acknowledged.

For more information go to the UniForum home page on the World Wide Web: http://www.uniforum.org or send inquiries to conferences@uniforum.org

ACM: Association for Computing Machinery

ASPLOS: Architectural Support for Programming Languages and Operating Systems

AUUG: Australian UNIX Systems Users Group

COOTS: Conference on Object-Oriented Technologies and Systems

DECUS: Digital Equipment Computer Users Society

EurOpen: European Forum for Open Systems

FIRST: Forum of Incident Response and Security Teams

GURU: Romanian UNIX User Group

GUUG: German UNIX Users Group

HotOS: Hot Topics in Operating Systems

IEEE: Institute of Electrical and Electronics Engineers

IETF: Internet Engineering Task Force

INET: Annual Conference of Internet Society

IWOOS: International Workshop on Object-orientation in Operating Systems

JUS: Japan UNIX Society

LISA: USENIX/SAGE Systems Administration Conference

OOPSLA: Object-oriented Programming Systems, Languages and Applications

OSDI: Symposium on Operating Systems Design & Implementation

POPL: Principles of Programming Languages

ROSE: Open Systems in Romania

SANS: System Administration, Networking & Security

SDNE: Services in Distributed and Networked Environments

SIGCOMM: Data Communication

SIGOPS: ACM Special Interest Group on Operating Systems

SIGPLAN: ACM Special Interest Group on Programming Languages

SIGSOFT: ACM Special Interest Group on Software Engineering

SOSP: ACM Symposium on Operating Systems Principles

SUG: Sun User Group

UKUUG: United Kingdom UNIX Systems Users Group

UniForum: International Association of UNIX and Open Systems Professionals

WITI: International Network of Women in Technology

CALENDAR OF EVENTS

This is a combined calendar of conferences, symposia, and standards meetings. If you ha an event that you wish to publicize, please contact *<login@usenix.org>*. For complete USENIX conference and symposia listings see URL *<http://www.usenix.org/events/general.html>*.

* = events sponsored by the USENIX Association.

1996

June

1 - 6 DECUS, '96, St. Louis, MO
3 - 4 SDNE '96, Estrada de Vitoria, Macau
5 - 7 WITI, Santa Clara, CA
10 -14 NetWorld+Interop '96, Frankfurt, Germany
17 - 21 *COOTS II, Toronto, Canada
24 - 28 INET '96, Montreal, Canada
24 - 28 IETF, Montreal, Canada

July

8 - 12 IEEE European Conference on Object-Oriented Programming, Berne, Switrzerland
10 - 13 *Tcl/Tk, Monterey, CA
14 - 19 IEEE POSIX, Nashua, NH
15 - 19 NetWorld+Interop '96, Tokyo, Japan
22 - 25 *6th UNIX Security, San Jose, CA
28 - 31 FIRST, Santa Clara, CA

August

4 - 9 SIGGRAPH, New Orleans, LA
5 - 9 Interex '96, San Diego, CA
28 - 30 SIGCOMM, Palo Alto, CA

September

3 - 5 GUUG, Leipzig, Germany
9 - 11 ACM SIGOPS European Workshop, Ireland
16 - 20 NetWorld+Interop '96, Atlanta, GA
18 - 20 AUUG, Melbourne, Australia
30 - Oct 4 * LISA '96, Chicago, IL

October

1 - 4 ASPLOS VII, Cambridge, MA
7 - 11 NetWorld+Interop '96, Paris, France
8 - 10 UNIX Expo, New York City
10 - 16 OOPSLA '96, San Jose, CA
23 - 25 IEEE Symposium on Reliable Distributed Systems, Niagara, Canada
27 - 28 *IWOOOS '96, Seattle, WA
28 - 31 *OSDI II, Seattle, WA
28 - Nov 1 NetWorld+Interop '96, London, England

November

4 - 8 Open Systems World/ FedUNIX Washington, DC
4 - 8 UNIX Network Security, Washington, DC
9 - 14 DECUS, Anaheim, CA
17 - 22 ACM IEEE-CS Supercomputing '96, Pittsburgh, PA
18 - 20* Electronic Commerce, Oakland, CA
25 - 29 NetWorld+Interop '96, Sydney, Australia

December

9 - 13 IETF, San Jose, CA
JUS UNIX Fair

1997

January

6 - 10 *USELINUX Conference, Anaheim,
6 - 10 *USENIX, Anaheim, CA
20 - 24 POPL '97

March

1 - 5 ACM '97, San Jose, CA
12 - 14 UniForum, San Francisco, CA

April

7 - 11 IETF, Memphis, TN
7 - 11 Networld+Interop '97, Singapore

May

5 - 7 HotOS-VI

June

16 - 20 SIGPLAN '97

October

12 - 17 OOPSLA '97
27 - 31 *LISA '97, San Diego, CA

1998

January

19 - 23 POPL '98

June

15 - 19 *USENIX, New Orleans, LA

December

7 - 11 *LISA '98, Boston, MA
JUS UNIX Fair

USENIX

1996 Conferences, Symposia, and Workshops for UNIX and Advanced Computing Systems Professionals

If these topics are important to you:

- **Tcl/Tk**
- **UNIX Security and Applications of Cryptography**
- **Object-Oriented Technology**
- **Systems Administration**
- **Operating Systems Design and Implementation**
- **Electronic Commerce**

then save these dates!

Plan to attend these USENIX events:

- **2nd Conference on Object-Oriented Technology.** June 17-21, 1996. Toronto
- **4th Annual Tcl/Tk Workshop.** July 10-13, 1996. Monterey, CA
- **6th UNIX Security Symposium Focusing on Applications of Cryptography.** July 22-25, 1996. San Jose, CA
- **10th Systems Administration Conference (LISA).** September 30-October 4, 1996. Chicago
- **2nd Symposium on Operating Systems Design and Implementation (OSDI).** October 28-31, 1996. Seattle, WA
- **2nd Electronic Commerce Workshop.** November 18–20, 1996. Oakland, CA

For more information:

USENIX, 22672 Lambert St., Suite 613,
Lake Forest, CA 92630
Phone: 714.588.8649
Fax: 714.588.9706
Email: *conference@usenix.org*
WWW: *http://www.usenix.org*

USENIX, the UNIX and Advanced Computing Systems Technical and Professional Association, offers technical conferences for and by technical professionals.
SAGE, the System Administrators Guild, a special technical group within USENIX, is dedicated to the advancement and recognition of system administration as a profession.

SECOND CLASS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street
Suite 215
Berkeley, CA 94710

POSTMASTER:
SEND ADDRESS CHANGES
TO *;login:*
USENIX ASSOCIATION
2560 NINTH STREET
SUITE 215
BERKELEY, CA 94710